WELCOME TO

iOS DEVELOPMENT

iOS DEVELOPMENT

ABOUT THIS COURSE

LOGISTICS

- Wednesdays, 5:30-8:30pm
- Ryerson 277
- Prerequisite: Core Programming or consent of instructor

LOGISTICS

- 2 Course Sections
  - Section 1: Tuesday/Thursday (daytime)
  - Section 2: Wednesday (evening)
- Shared course website, office hours, Slack
- You must attend the section for which you are registered

INTRODUCTIONS

- Andrew Binkowski (Section 1 Instructor)
- Susan Stevens (Section 2 Instructor)

INTRODUCTIONS

- Hannah Bennett (TA)
- Sammy Cannillo (TA)
- Pero Atanasov (Grader)

WHAT YOU'LL NEED

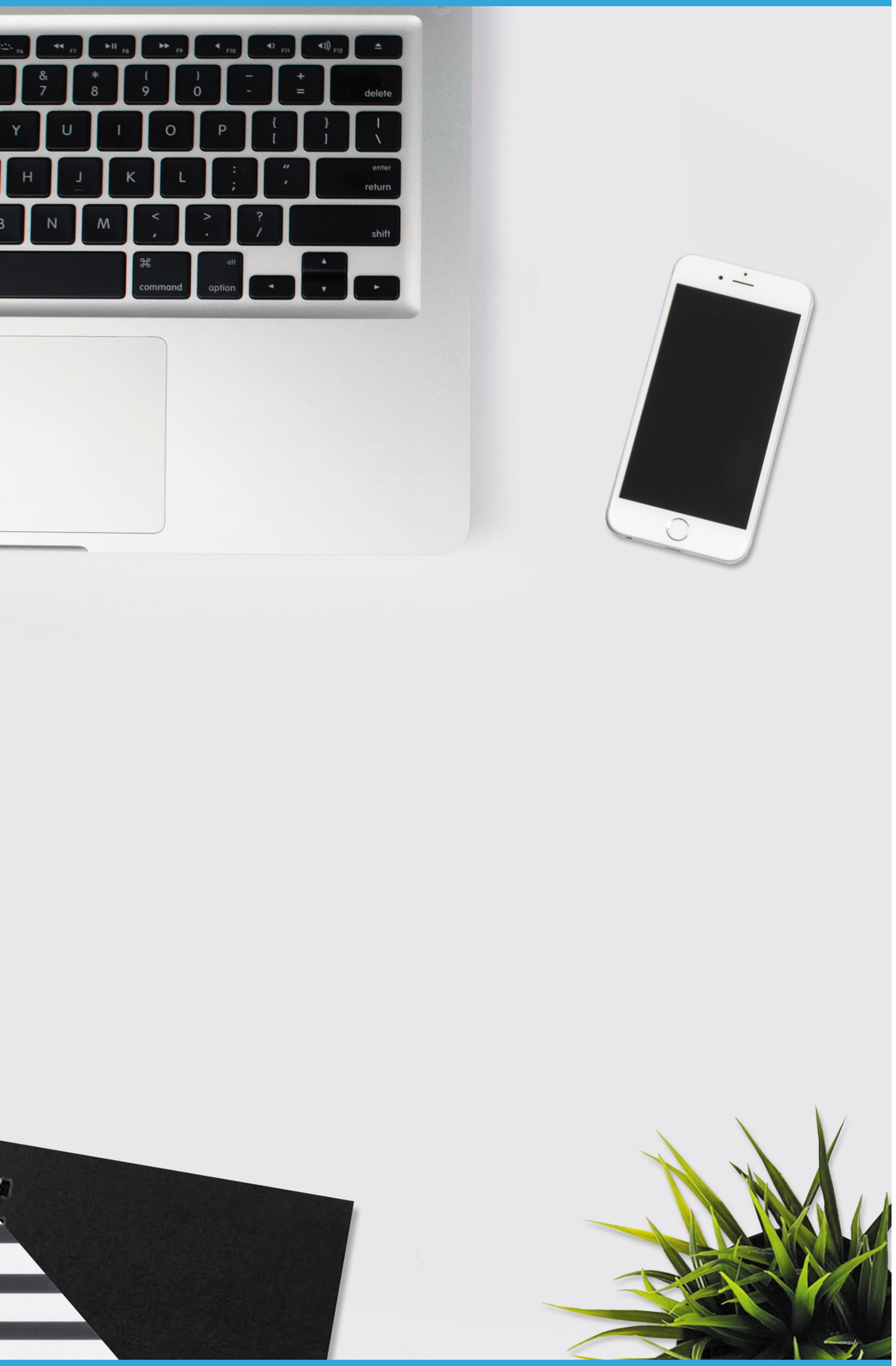
- Xcode 11
  - Available for free on the Mac App Store
  - Requires a Mac running Mojave or Catalina
- Access to an iOS device (optional)
  - Nice to have, but not required

WHAT YOU'LL NEED

- Apple Developer account
    - Free or paid ($99 per year)
- Start with a free account (upgrade later if you want)
- Apple requires a paid account to submit apps to the App Store

WHERE THIS COURSE COULD TAKE YOU

- Career in iOS development
- Publish apps in the App Store
- Scholarship to WWDC



iOS DEVELOPMENT

COURSEWORK

HOMEWORK ASSIGNMENTS

- Build one app per week
  - Small, but fully-functioning
  - Due @ 5:29pm before the next week's class
- Graded on functionality, not aesthetics

HOMEWORK ASSIGNMENTS

- Must compile with no errors or warnings
- Include links to all sources used in a comment above the relevant code
- Include an app icon

HOMEWORK ASSIGNMENTS

- GitHub Classroom
  - Each week, you will get a link to create a private repo
  - To submit your assignment, open a pull request from `development` into `master`

CASE STUDIES

- Pick an app currently in the App Store
    - What role does it play in the developer's overall business model?
    - What's the competition like? How does the app set itself apart?
    - Technical analysis. Explain how the key features of the app may have been built.

CASE STUDIES

- Work individually or in pairs
- 5 minute presentations
- Choose an app you love 💕

MAKING AN APP IS EASY. MAKING A REALLY GOOD APP IS HARD.

FINAL PROJECTS

- Build an App Store-worthy app of your own design
    - 3 weeks to complete
    - Presentations will be held during finals week

DUE DATES

- Assignment 1: Wed, Jan 15
- Assignment 2: Wed, Jan 22
- Assignment 3: Wed, Jan 29
- Assignment 4: Wed, Feb 5
- Assignment 5: Web, Feb 12
- Assignment 6: Wed, Feb 19
- Assignment 7: Wed, Feb 26
- Case studies: Wed, Mar 4
- Final Projects: Wed, Mar 18

LATE WORK POLICY

- Weekly homework assignments are due at 5:29 pm on Wednesdays
- You may use **one** extension
  - Deadline extended to midnight on Friday
  - Send me an email or slack message before class on Wednesday
- No extensions for case studies or final projects

GRADING

- 70% - Homework assignments (10% each)
- 5% - Case study presentation
- 25% - Final project

ACADEMIC HONESTY POLICY

- All code submitted must be your own work
- You may discuss assignments with other students, but **you may not share code**
- You may use online sources as long as you **provide a link** to the source above the relevant code

iOS DEVELOPMENT

RESOURCES

WHERE TO GET HELP

- Course website
  - http://uchicago.mobi
- Sample code and slides
  - https://github.com/uchicago-mobi/mpcs-51030-winter-2020-section-2
- Office hours
- Slack

OFFICE HOURS

	Day	Time	Location
Susan	Saturday	2-4 PM	Hyde Park (JCL 352)
Hannah	Sunday	1:30-3:30 PM	Harold Washington Library
Sammy	Monday	6-8 PM	Gleacher Center
Andrew	Tuesday	12:30-1:30 PM	Hyde Park
Andrew	Thursday	10-11 AM	Hyde Park

SLACK GUIDELINES

- Post questions about assignments and course material in Slack.
- Use threads. This will make it easier for others to read the conversation later.
- Feel free to answer questions from other students.
- Post brief code snippets, not entire files.

SLACK GUIDELINES

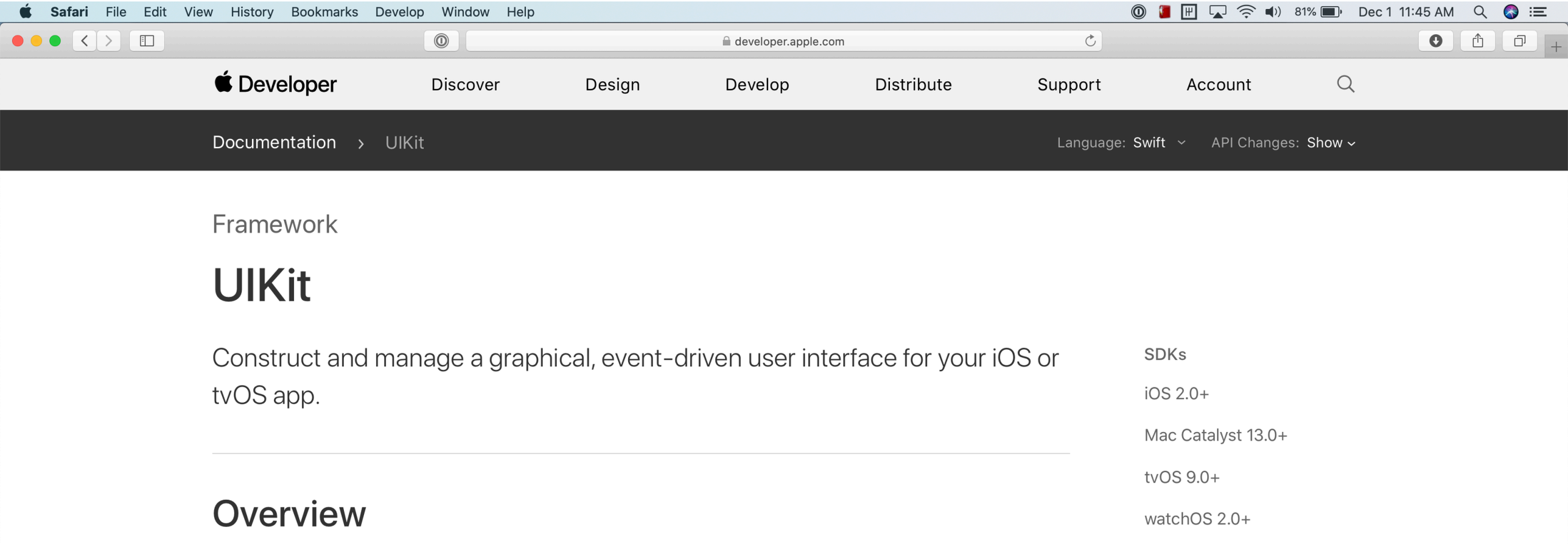
- Keep an eye on Slack for:
  - Changes to office hour times and locations
  - Clarifications on assignments
  - General announcements

NO TEXTBOOK

- iOS development changes quickly
  - Upcoming changes announced in June at WWDC
  - New version of iOS released in September
- Book publishers can’t always keep up. However, there are lots of good resources online.

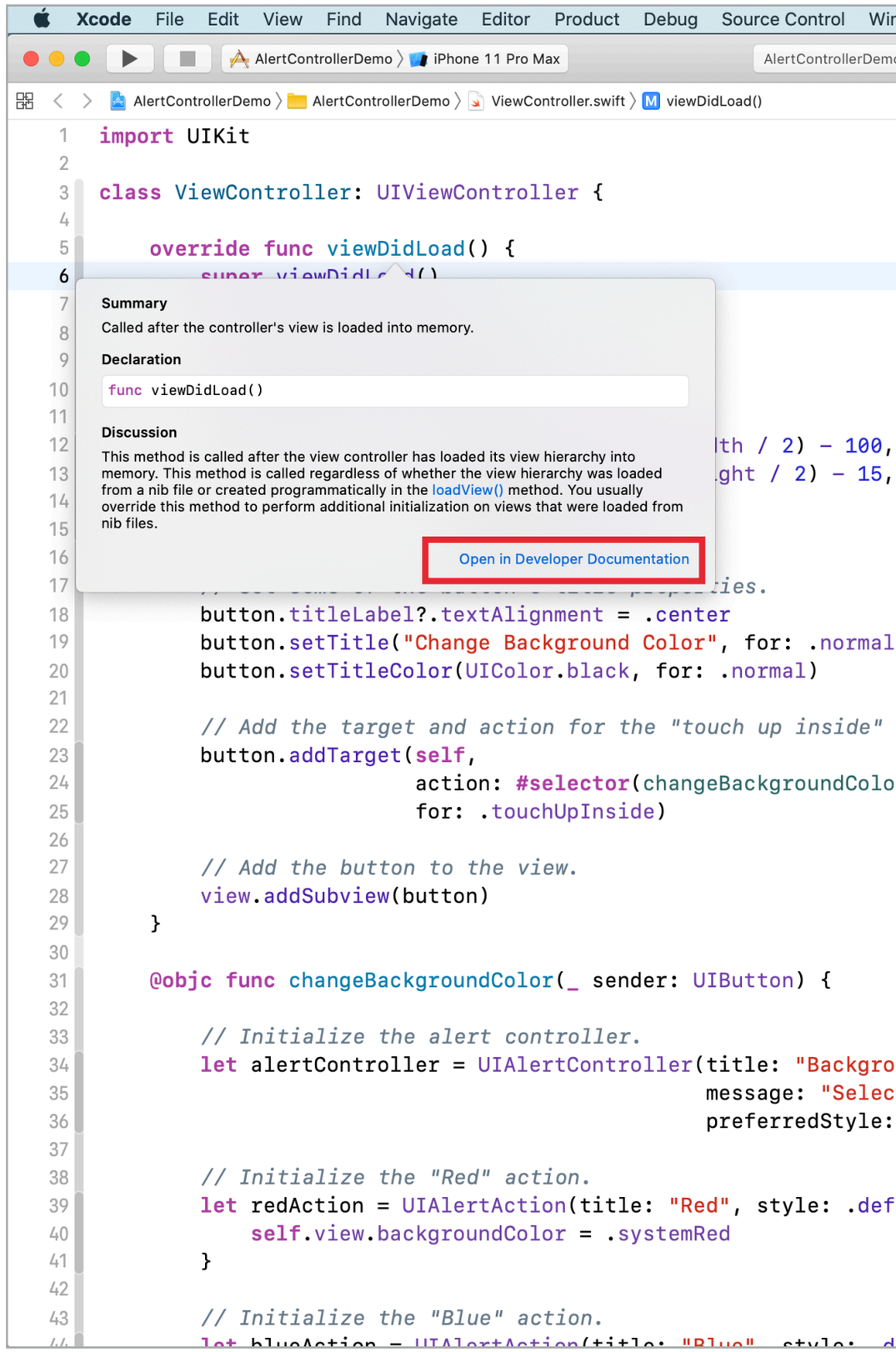
ONLINE RESOURCES

- Apple Documentation
  - https://developer.apple.com/documtation



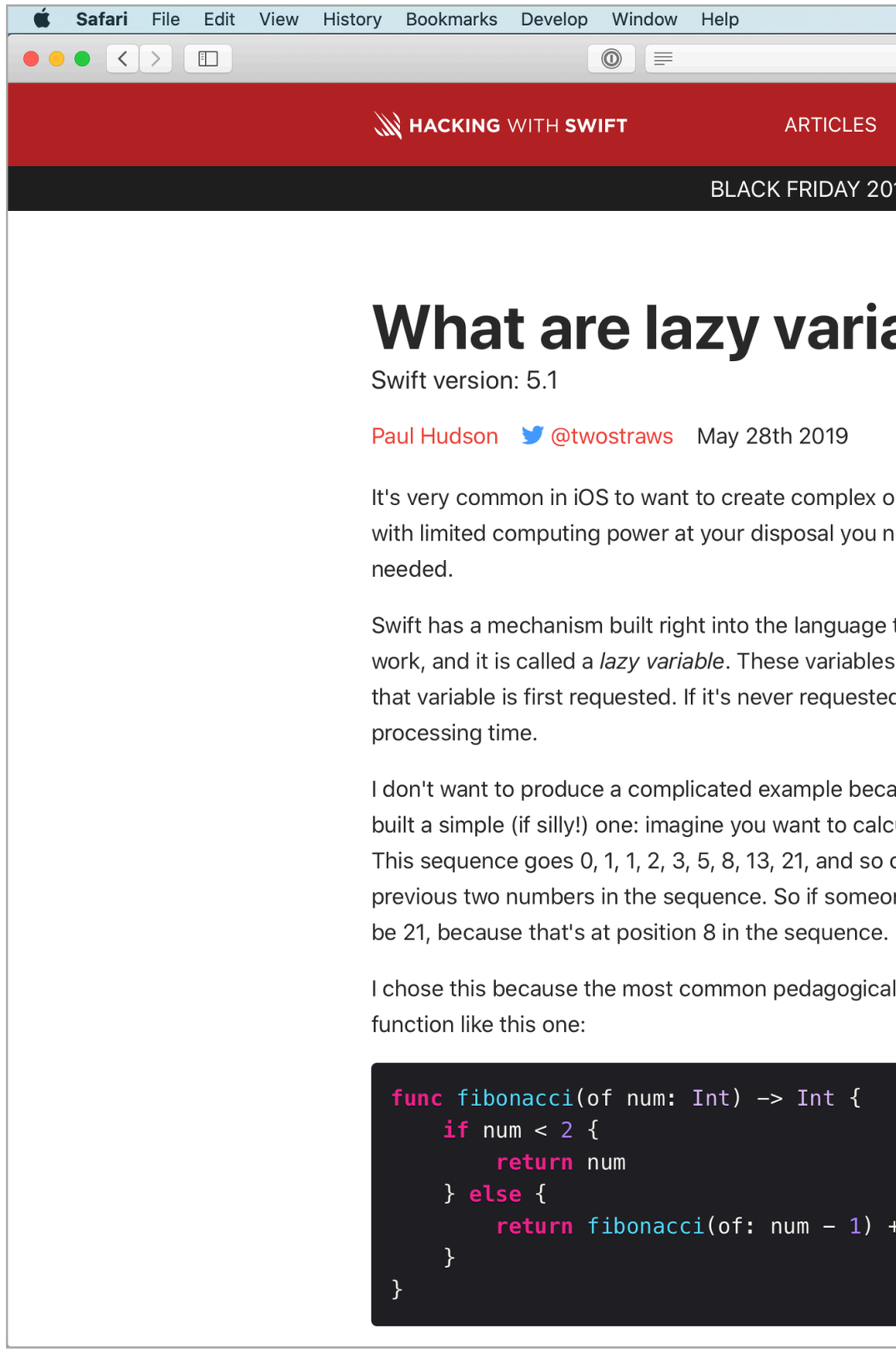
ONLINE RESOURCES

- Apple Documentation
  - In Xcode, Option+Click on a function or type to preview the documentation
  - Click Open in Developer Documentation to open the full documentation



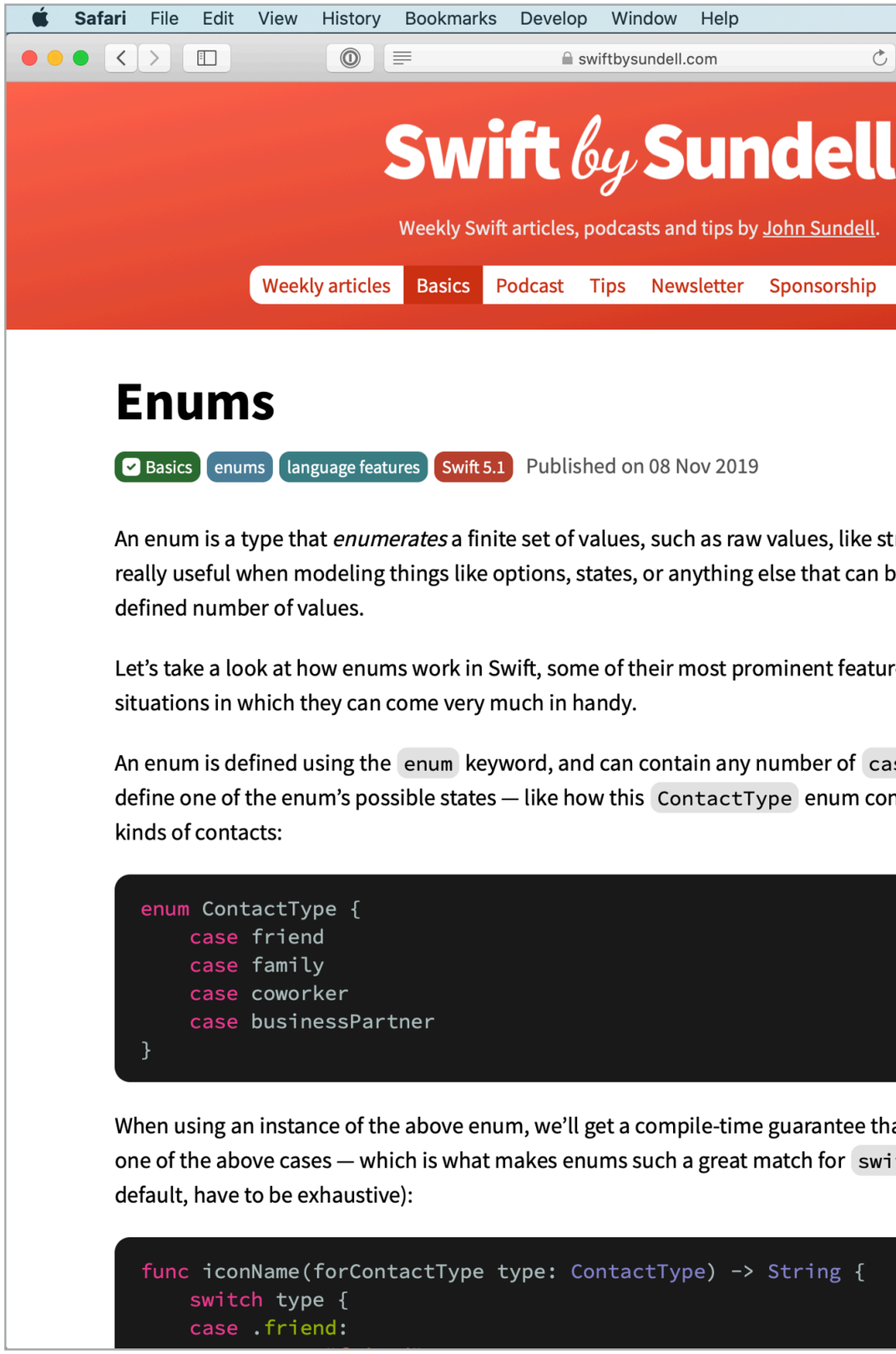
ONLINE RESOURCES

- Hacking with Swift
  - Short articles, videos and code snippets
- Good for quick help on a wide variety of topics



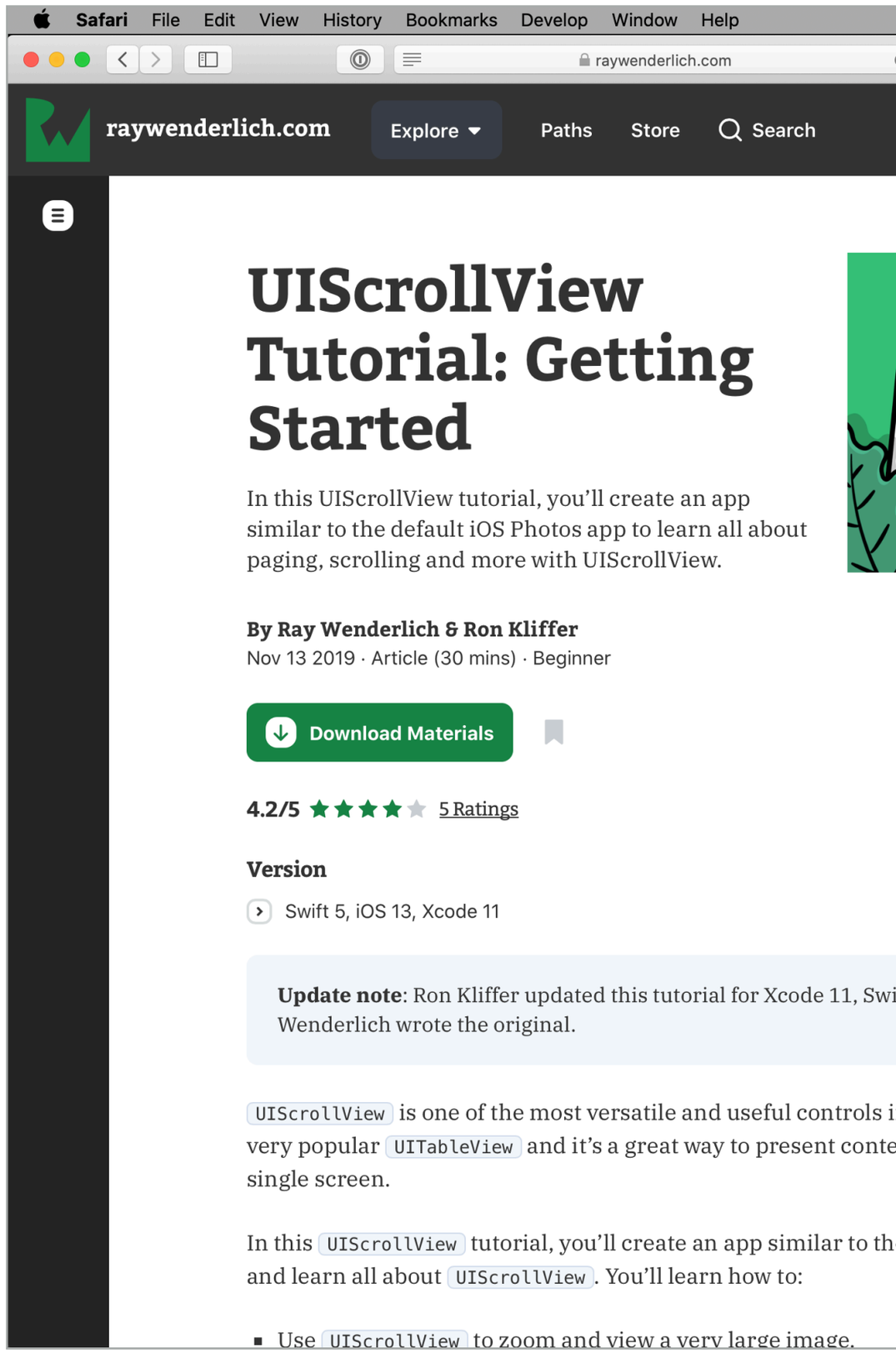
ONLINE RESOURCES

- Swift by Sundell
  - In-depth articles on Swift best practices and iOS patterns



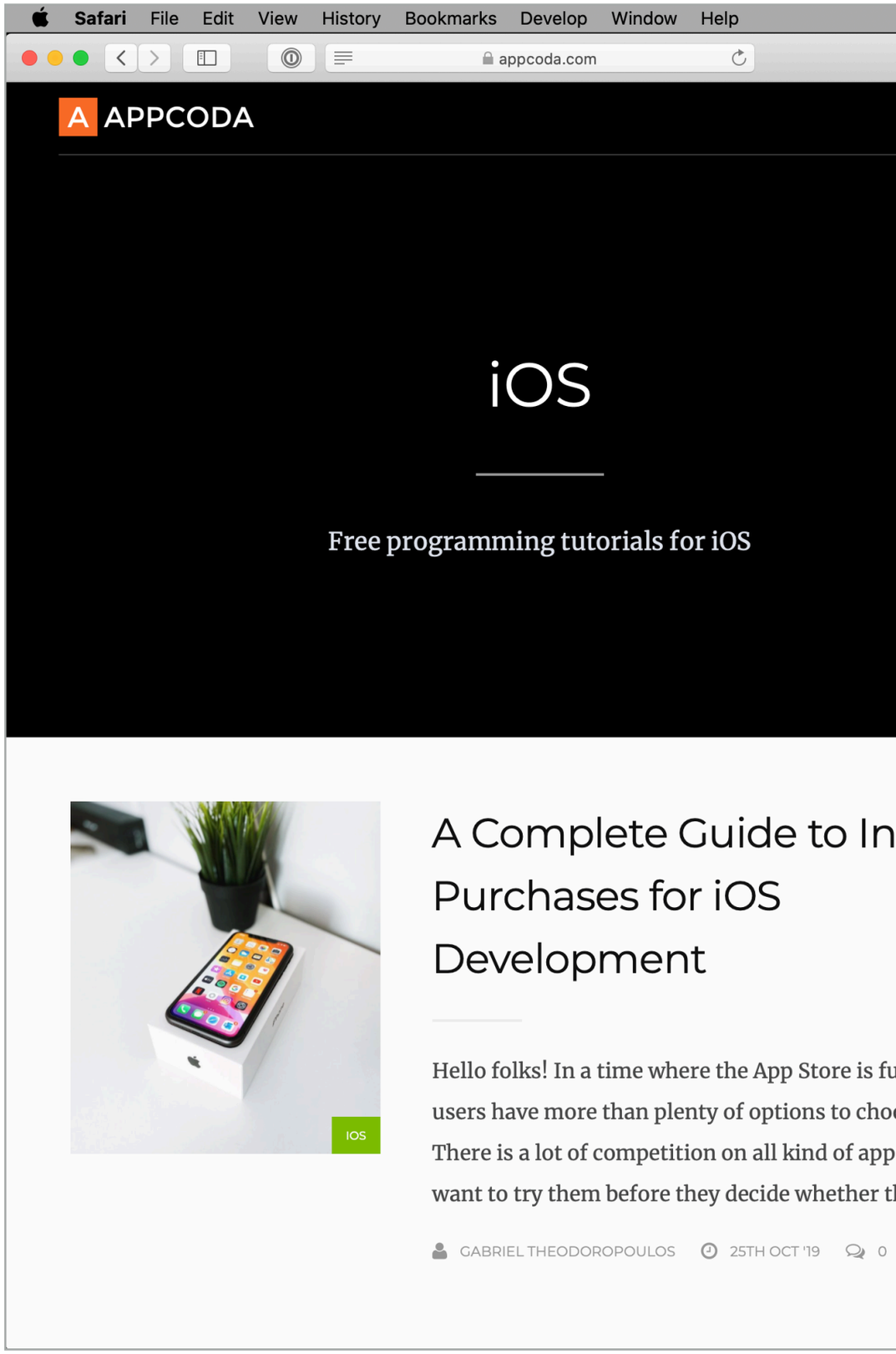
ONLINE RESOURCES

- Ray Wenderlich
  - Tutorials that walk you through creating an app from start to finish
- Articles are free, videos require subscription



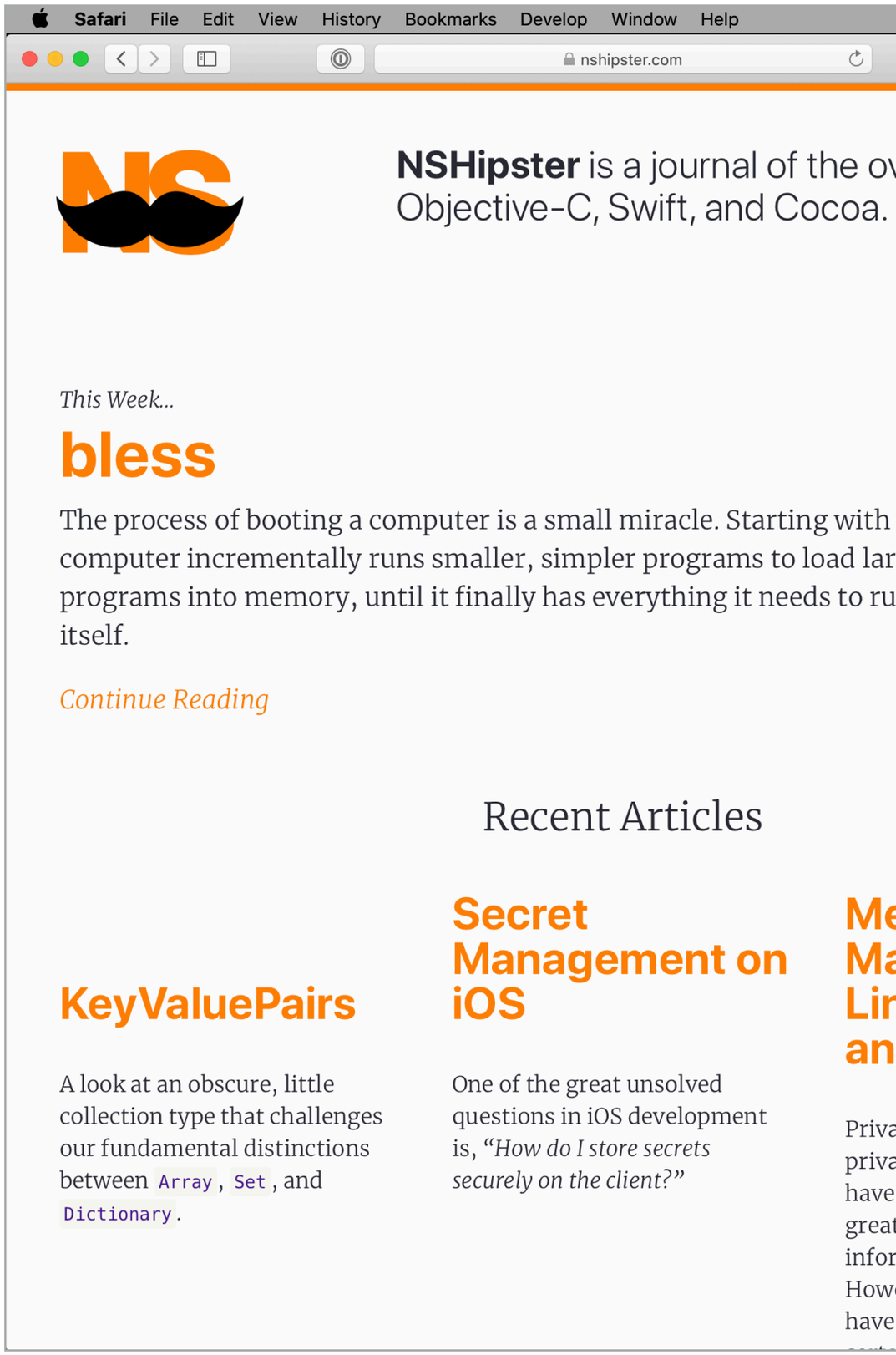
ONLINE RESOURCES

- AppCoda
  - Longer articles and tutorials on iOS and Swift



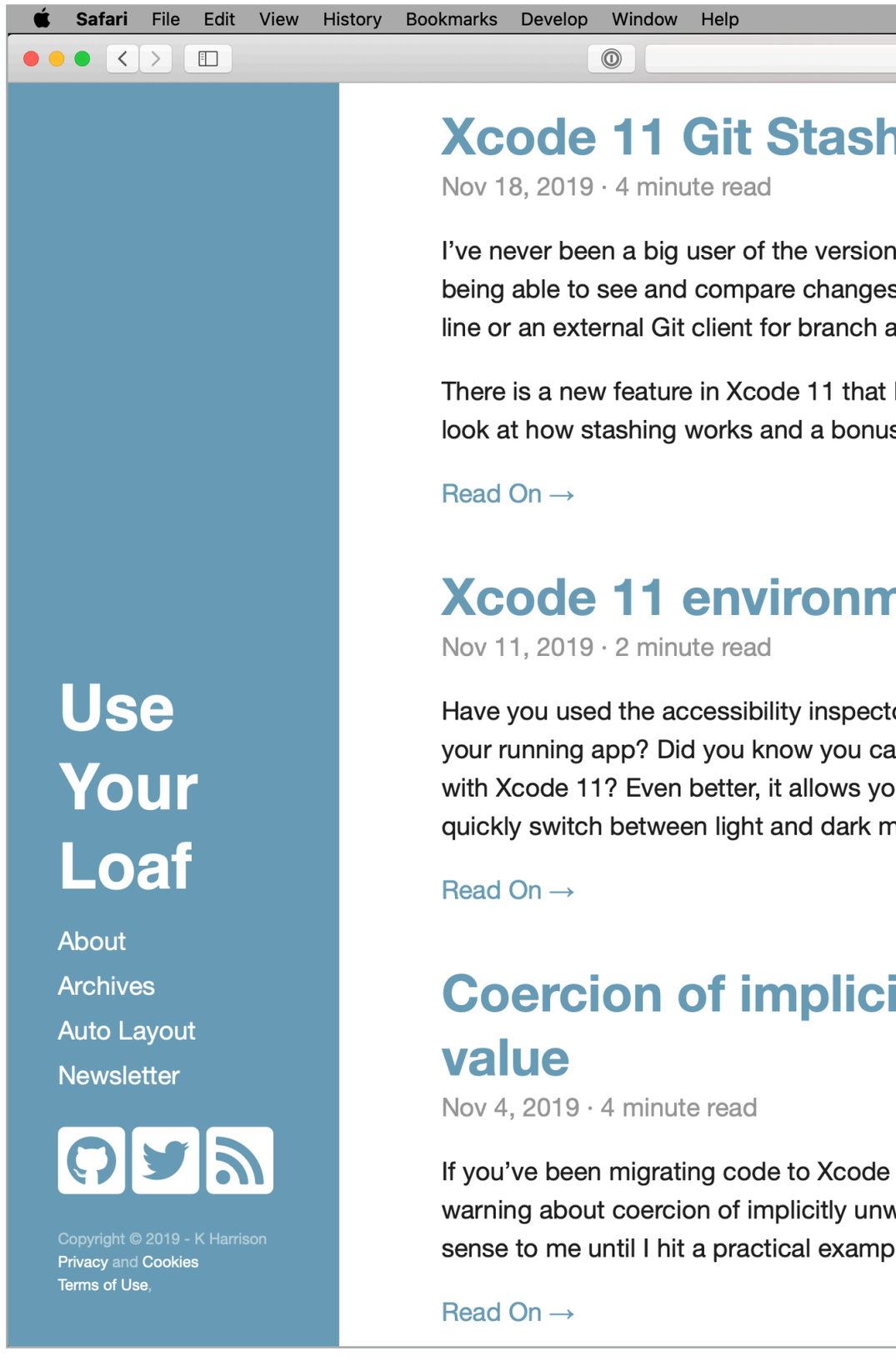
ONLINE RESOURCES

- NSHipster
  - Behind-the-scenes looks at various Swift, Objective C and iOS topics



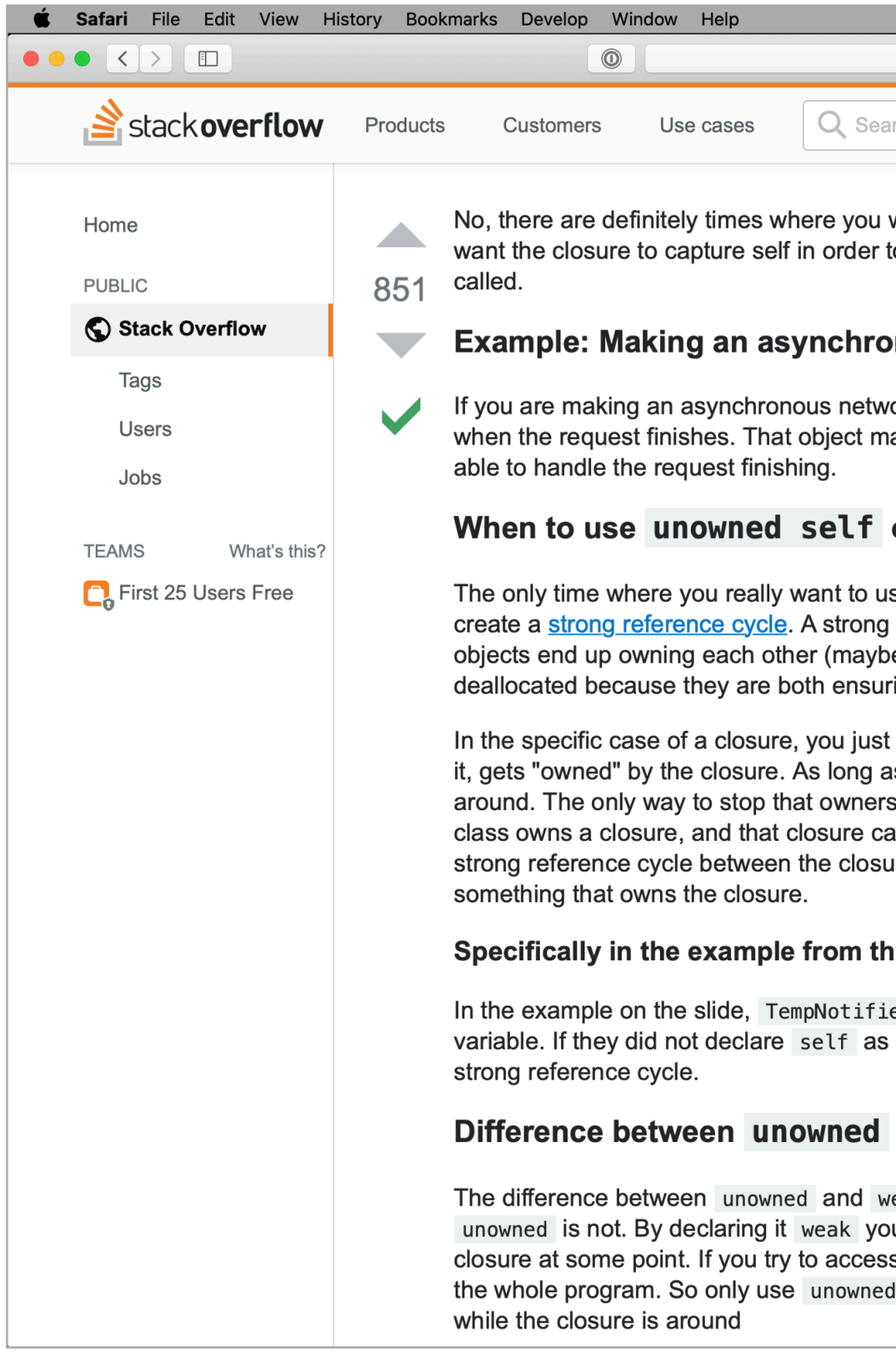
ONLINE RESOURCES

- Use Your Loaf
  - Short, easy-to-understand articles on Swift, Xcode and iOS



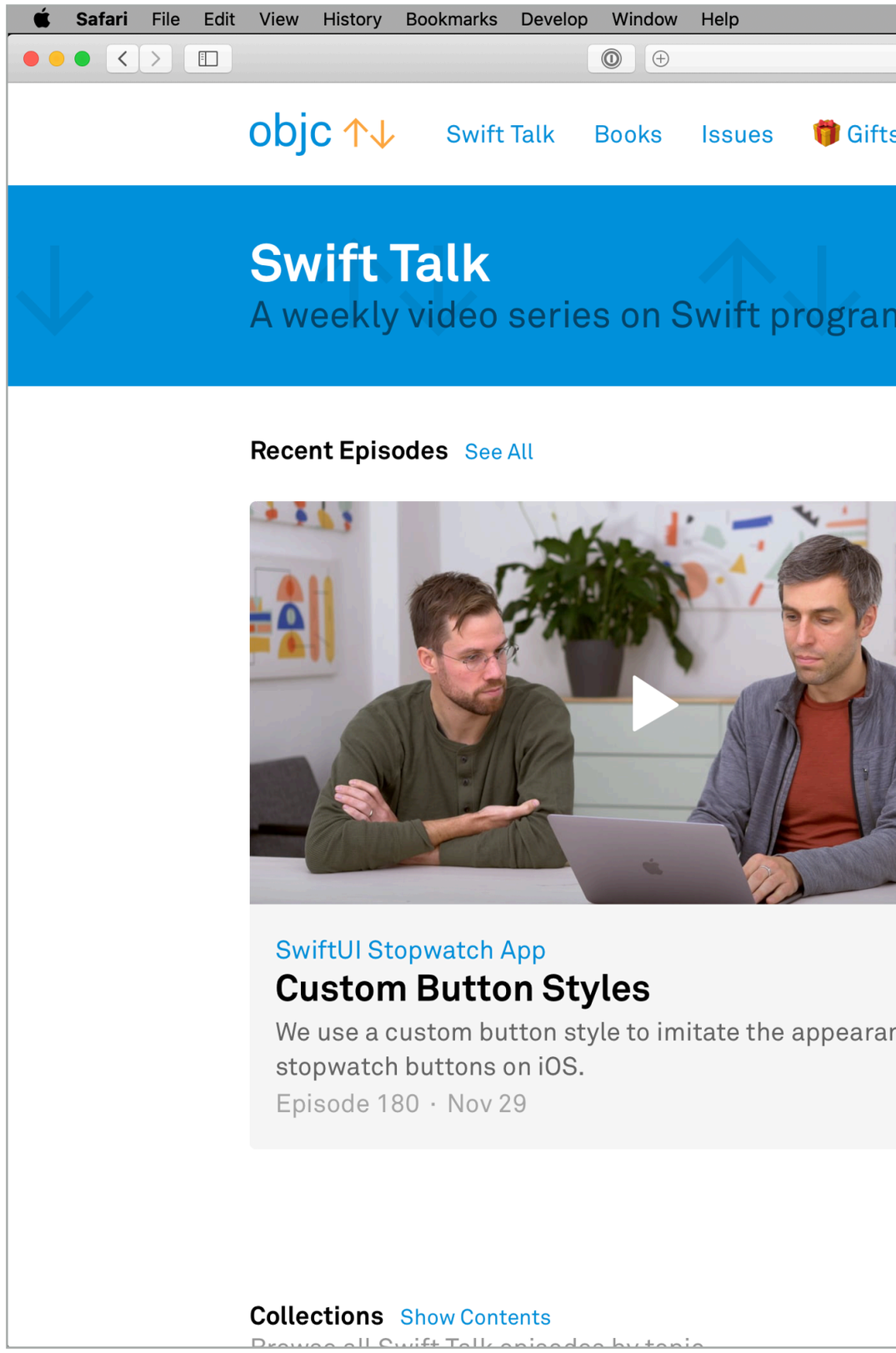
ONLINE RESOURCES

- Stack Overflow
  - Answers to (almost) every question imaginable



ONLINE RESOURCES

- objc.io
  - Weekly videos with live coding
- Some free, most require a subscription



A NOTE OF CAUTION

- iOS development changes quickly
    - Check the date on whatever you're reading
    - Apple Documentation is the source of truth
    - If you use older syntax, Xcode will try to help you update it

THE CHANGING LANDSCAPE OF

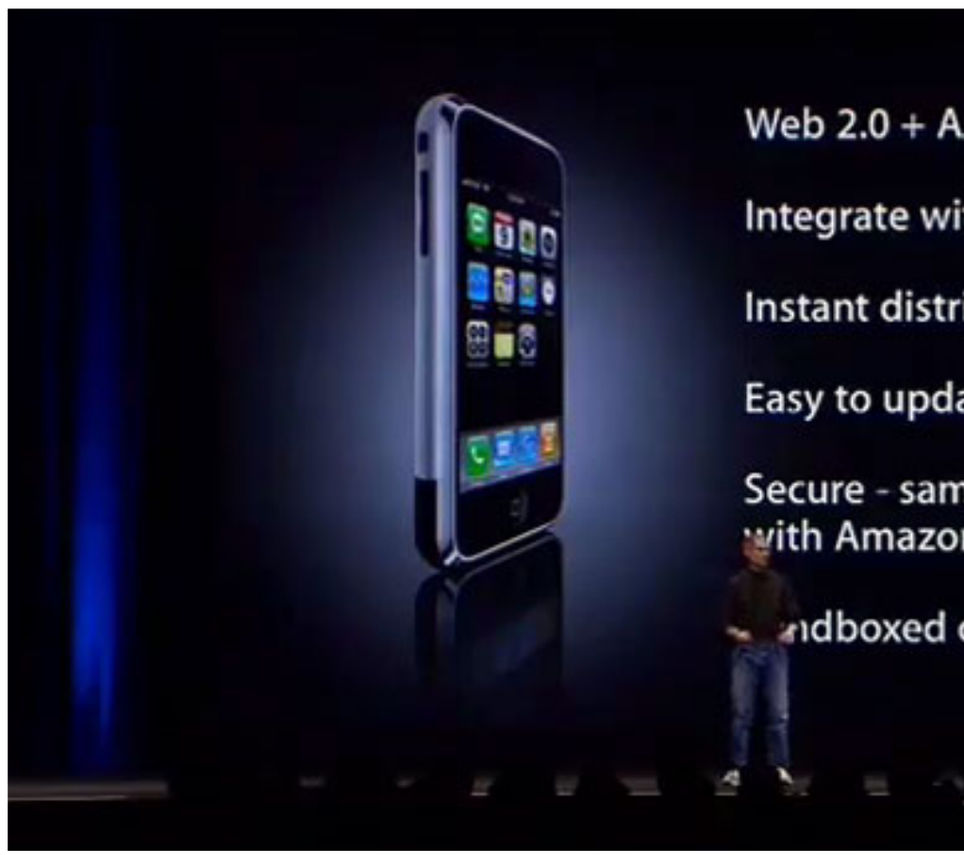
iOS DEVELOPMENT

iPhone OS 1 (2007)

- First iPhone released
- No App Store
- “What about developers? ...We've come up with a very sweet solution.” – Steve Jobs, WWDC 2007

iPhone OS 1 (2007)

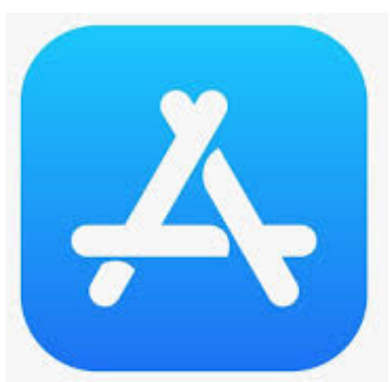
- Problems with the “sweet solution”
  - No access to the device’s core functionality
  - Less performant than native apps
- How do you make money?



iPhone OS 2 (2008)

- iPhone SDK announced in March 2008
- App Store opened for business in July 2008
- 500 apps available
- Apple's advantage? iTunes
- 70/30 split between developers and Apple

+



iPhone OS 3 (2009)

- In-App Purchases & “freemium” apps
- Expanded home screen (up to 180 apps)

iOS 4 (2010)

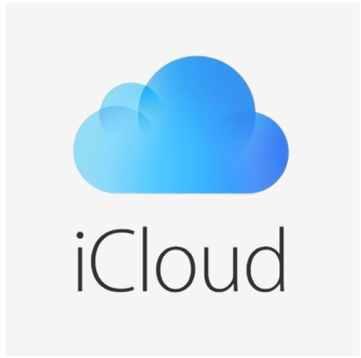
- iPhone OS renamed to iOS
- Folders (up to 2160 apps)
- Multitasking

YEARLY DEVELOPMENT CYCLE

- Each year, Apple announces changes to the SDK at the World Wide Developer Conference (WWDC)
  - WWDC is held in June
  - Beta versions released throughout the summer
  - New version of iOS released in September

NEW FRAMEWORKS

- iCloud (iOS 5)
- SpriteKit (iOS 7)
- HealthKit (iOS 8)
- WatchKit (iOS 8)
- HomeKit (iOS 8)
- SiriKit (iOS 10)
- ARKit (iOS 11)

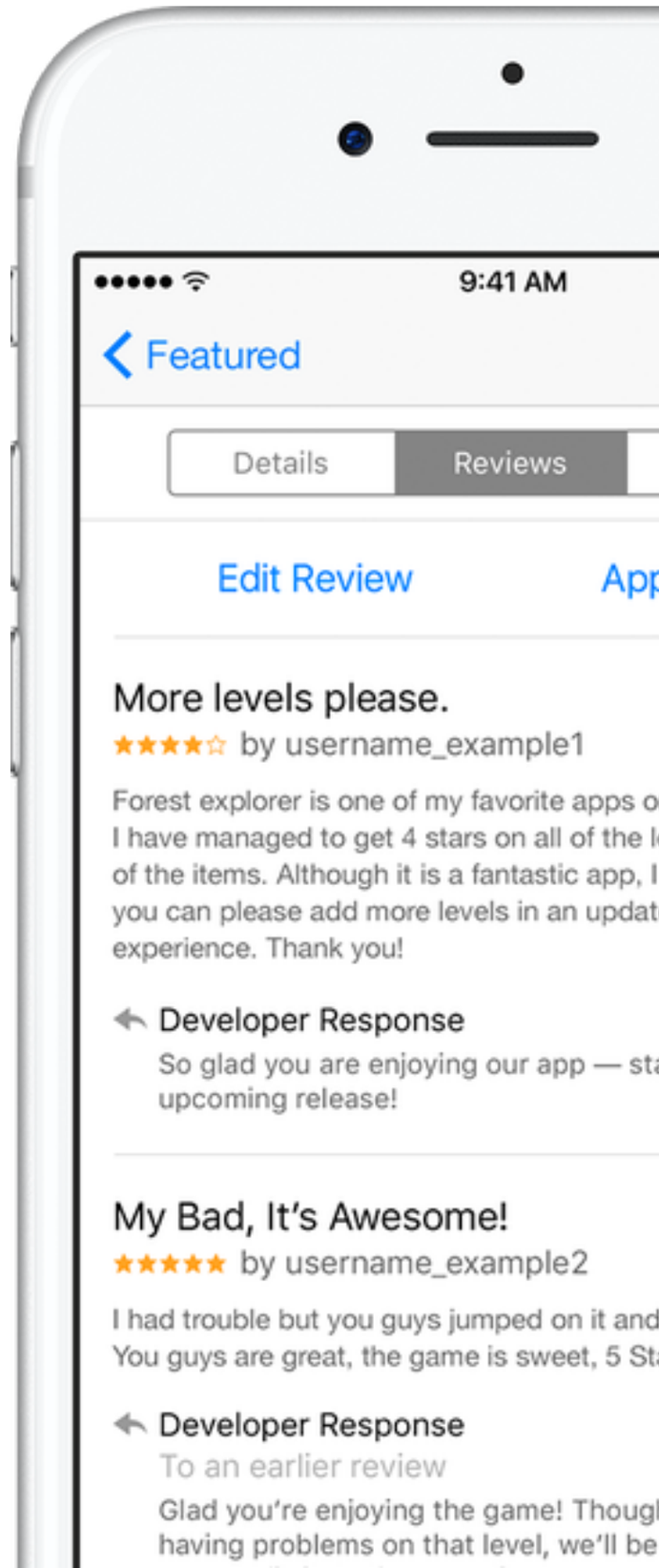


UI CHANGES

- iOS 7 redesign
- The “notch” introduced with iPhone X

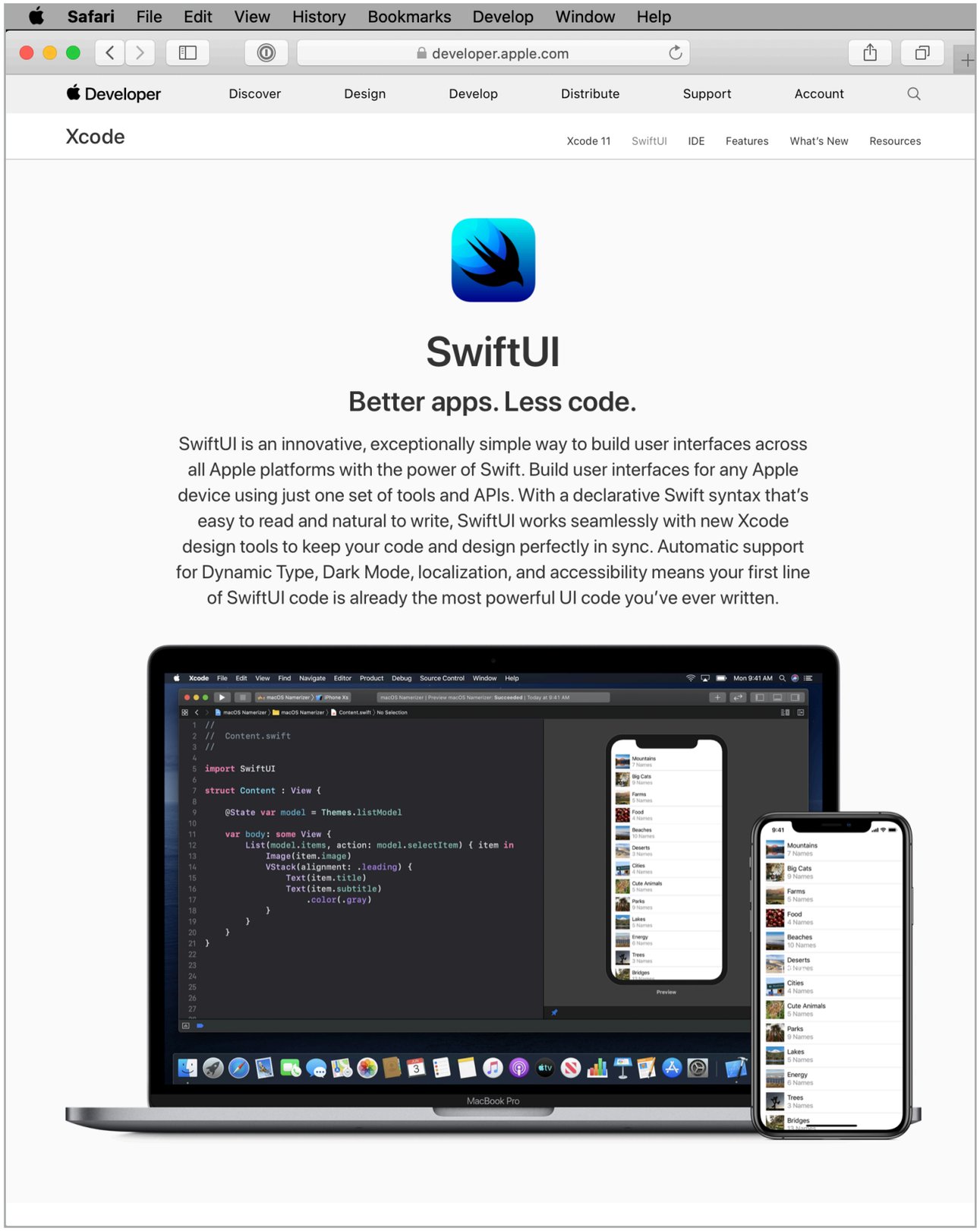
APP STORE CHANGES

- Automatic app updates (iOS 7)
- App thinning (iOS 9)
- Updates to ratings and reviews (iOS 11)
  - Submit reviews within the app
  - Developers can respond to reviews



WWDC 2019

- Big announcements:
  - Catalyst
  - SwiftUI



SWIFTUI

- SwiftUI is Apple's new declarative UI framework
  - For most of the quarter, we will use UIKit, the "traditional" way of building user interfaces
- We will discuss SwiftUI towards the end of the quarter

ContentView.swift

```swift
import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            MapView()
                .frame(height: 300)

            CircleImage()
                .offset(y: -130)
                .padding(.bottom, -130)

            VStack(alignment: .leading) {
                Text("Turtle Rock")
                    .font(.title)
                HStack(alignment: .top) {
                    Text("Joshua Tree National Park")
                        .font(.subheadline)
                    Spacer()
                    Text("California")
                        .font(.subheadline)
                }
            }
            .padding()
        }
    }
}

struct ContentView_Preview: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

AN INTRODUCTION TO

SWIFT

OBJECTIVE C

- Objective C was the language of Mac and iOS app development for many years
  - Developed in the early 1980's
  - Originally owned by Stepstone
  - Sold to NeXT in 1995
  - Apple acquired NeXT in 1996

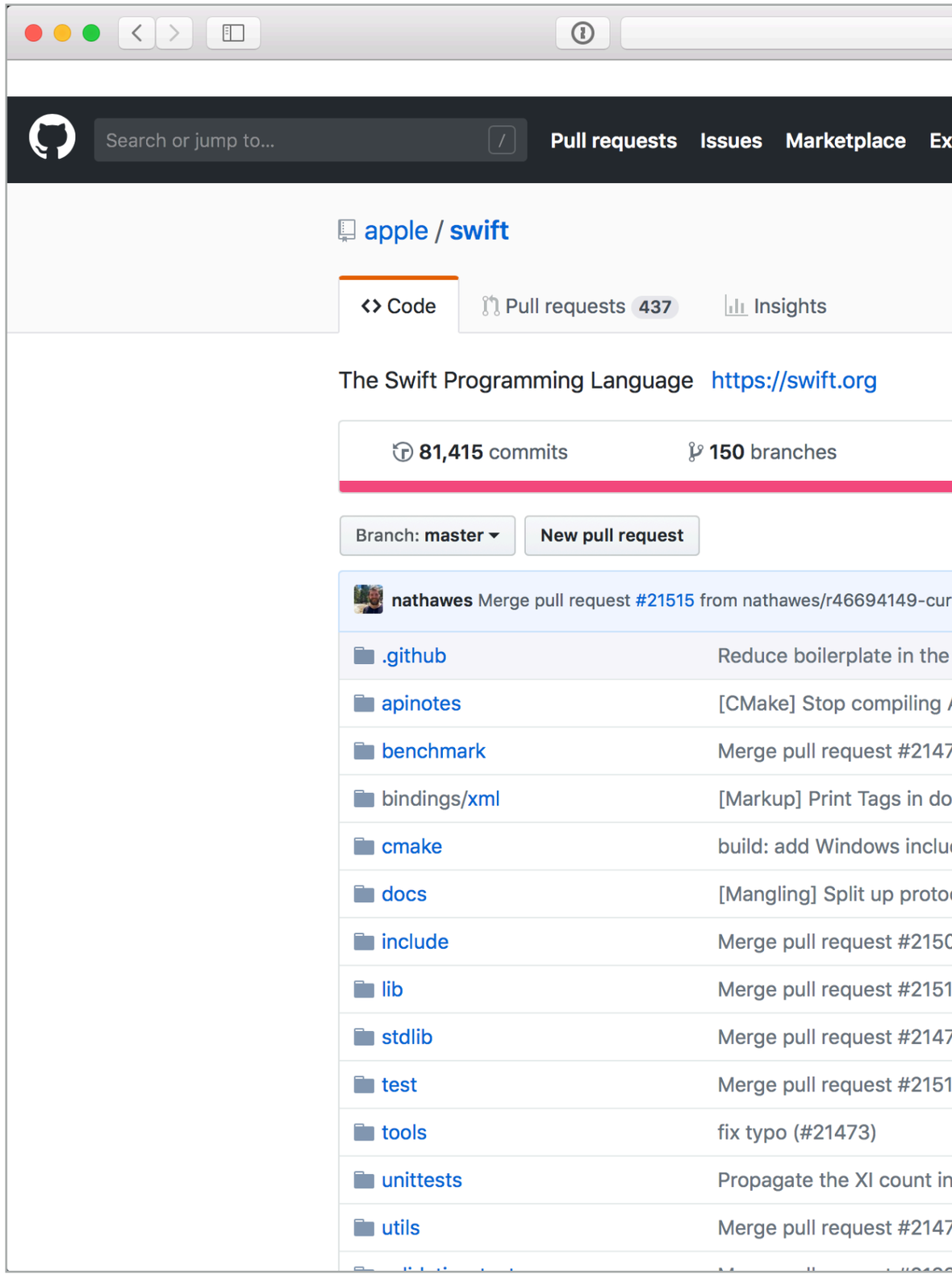
OBJECTIVE C

- Built on the C programming language
- Focused on “objects” rather than “procedures”

```
9  #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @end
14
15 @implementation ViewController
16
17 - (void)viewDidLoad {
18     [super viewDidLoad];
19     [self.view setBackgroundColor:UIColor.lightGrayColor];
20 }
21
22 @end
23
```

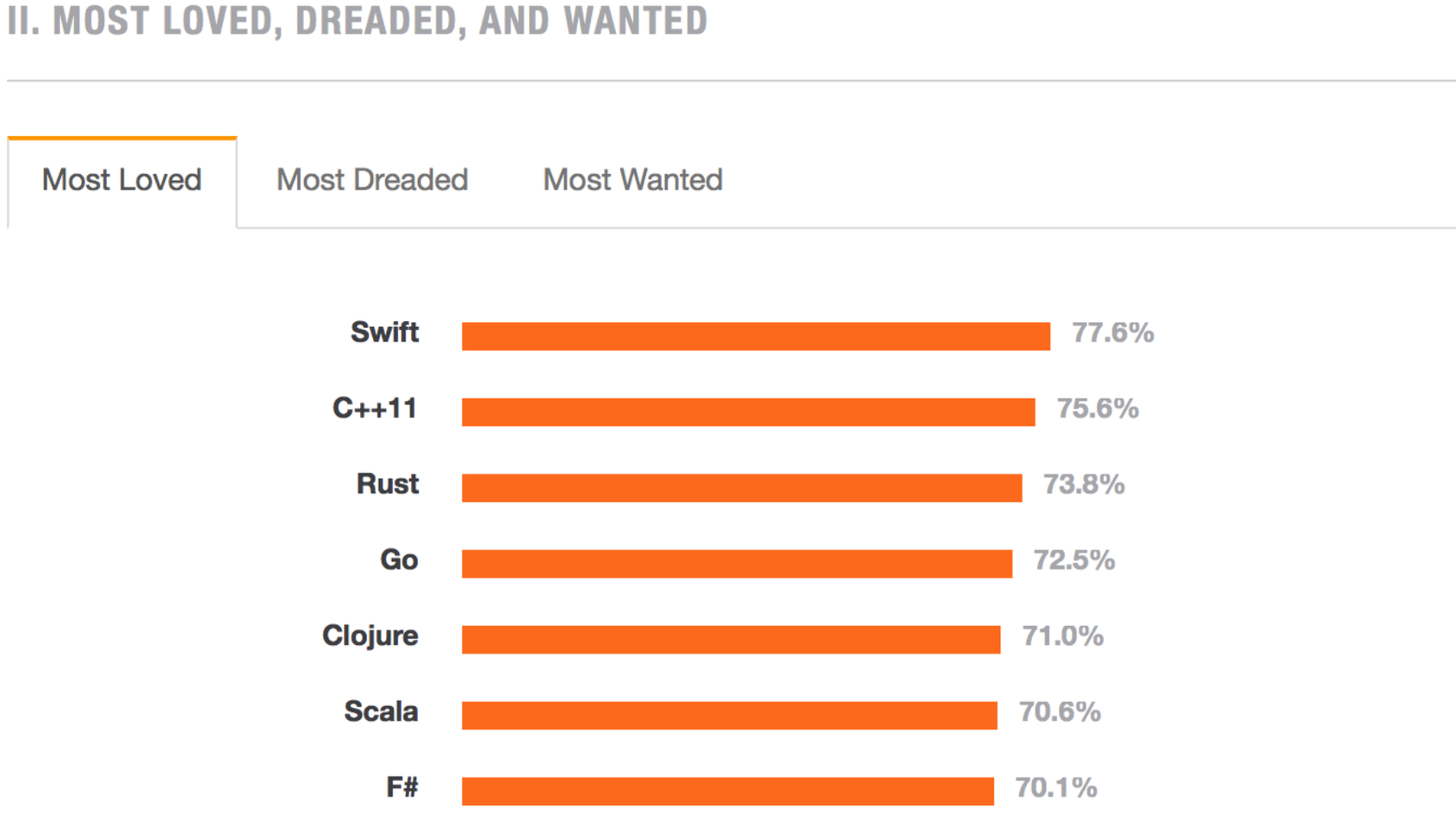
SWIFT

- "Objective C without the C"
- Introduced at WWDC in 2014
- Open-sourced in 2015



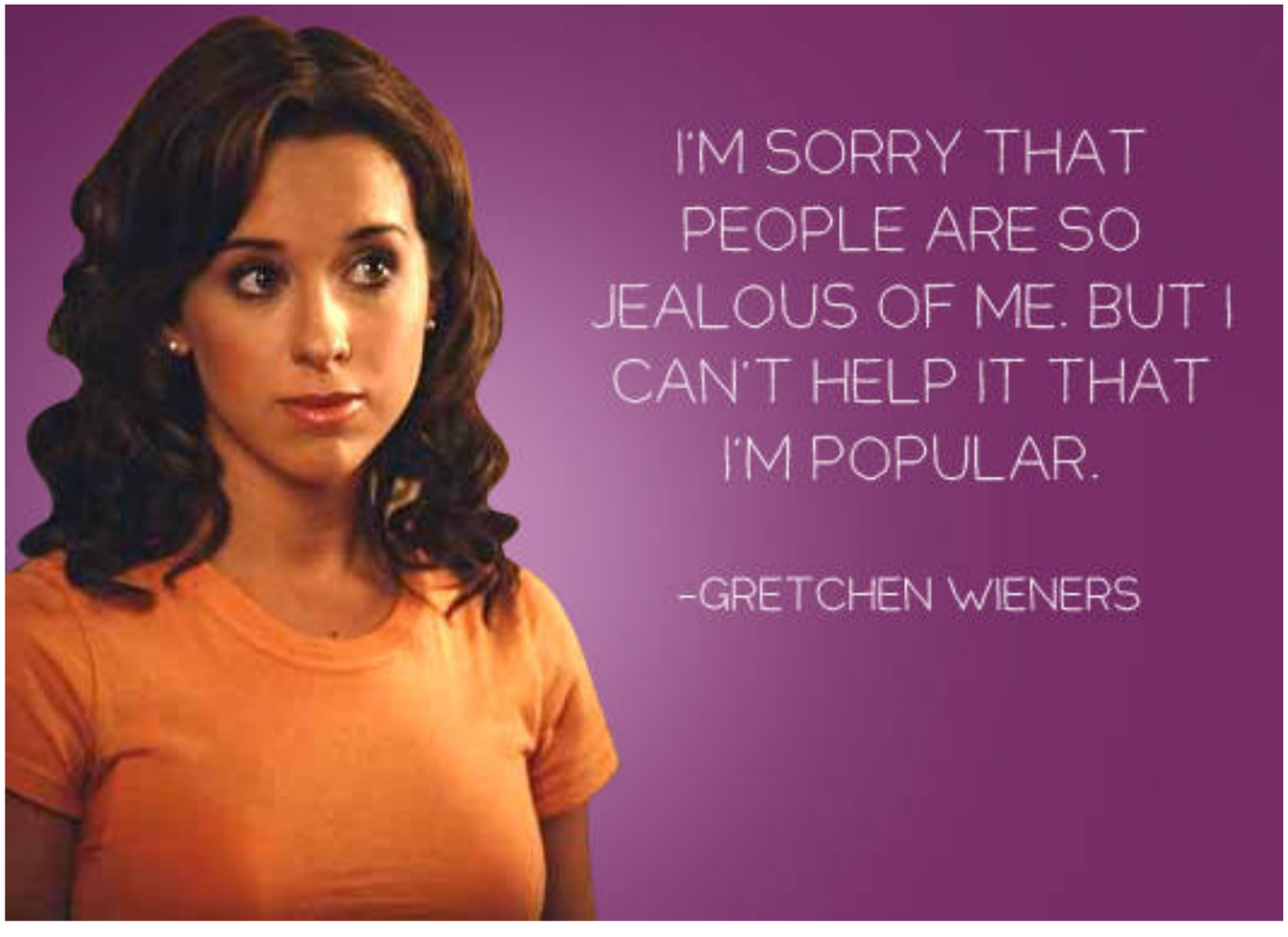
POPULARITY

- Most loved programming language on StackOverflow's 2015 Developer Survey



POPULARITY

- Tied with Objective C in January 2018 on RedMonk's Programming Language Rankings



GOALS OF SWIFT

- Swift aims to be:
  - Safe (protect against developer mistakes)
  - Fast (matching the performance of C-based languages)
  - Expressive (syntax that is a joy to use)

SWIFT ROADMAP

- **Variables**
- Control Flow
- Functions
- Classes
- Structs
- Enums
- Optionals

VARIABLES

```
var myVariable = 42
myVariable = 50
```

- Variables are declared using the `var` keyword
- Variables are mutable, meaning their values can be changed

CONSTANTS

```
let myConstant = 42
```

- Constants are declared using the `let` keyword
- Constants are immutable, meaning their value never changes
- What happens if you try to change the value of a constant?

TYPE INFERENCE

```
let implicitInteger = 70
let implicitDouble = 70.0
let explicitDouble: Double = 70
```

- The compiler will infer a type whenever possible
- You can specify an explicit type when necessary

TYPE SAFETY

```
var count = 3
count = "four" // compiler error
```

- Once a variable's type has been declared, its type cannot change
- Why is this called type safety?

SWIFT ROADMAP

- Variables
- **Control Flow**
- Functions
- Classes
- Structs
- Enums
- Optionals

IF STATEMENTS

```
if score > 10 {
    print("Outstanding")
} else if score > 0 {
    print("Nice work")
} else {
    print("Better luck next time")
}
```

- Parentheses around the condition are optional
- Curly braces around the body are required

SWITCH STATEMENTS

```
let soup = "tomato"

switch soup {
    case "minestrone":
        addTopping("parmesan cheese")
    case "tomato":
        addTopping("goldfish")
    default:
        addTopping("parsley")
}
```

- Use `switch` to compare a value to multiple cases
- Switch statements **do not** fall-through by default

FOR LOOPS

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0

for score in individualScores {
    if score > 50 {
        teamScore += 3
    } else {
        teamScore += 1
    }
}
```

- Use `for <variable> in <array>` to iterate through an array

FOR LOOPS

```
let ingredients = ["flour", "eggs", “sugar"]

for (index, ingredient) in ingredients.enumerated() {
    print("\(index + 1). \(ingredient)")
}
```

- Use the `enumerated` method if you need the current index as well as the value in the array

FOR LOOPS

```
var total = 0

for i in 0..<4 {
    total += i
}
```

- Use `..<` to create a range of integers

WHILE LOOPS

```
var n = 2

while n < 100 {
    n *= 2
}
```

- Use `while` to repeat a block of code (`n *= 2`) until a condition (`n < 100`) is no longer true

SWIFT ROADMAP

- Variables
- Control Flow
- **Functions**
- Classes
- Structs
- Enums
- Optionals

FUNCTIONS

```
5  func greet(name: String, hometown: String) -> String {
6      return "Welcome \(name), of \(hometown)"
7  }
8
9  greet(name: "Homer Simpson", hometown: "Springfield")
10
```

- Declared with the `func` keyword
- Return type comes after the `->` arrow

FUNCTIONS – ARGUMENT LABEL VS PARAMETER NAME

```
12  func greet(name: String, from hometown: String) -> String {
13      return "Welcome \(name), of \(hometown)"   << Internal
14  }
15
16  greet(name: "Homer Simpson", from: "Springfield")  << External
17
```

- The argument label (`from`) is used *externally*
- The parameter name (`hometown`) is used *internally*
- Note that `name` is both an argument label and a parameter name

FUNCTIONS – OMITTING ARGUMENT LABELS

```
19  func greet(_ name: String, from hometown: String) -> String {
20      return "Welcome \(name), of \(hometown)"
21  }
22
23  greet("Homer Simpson", from: "Springfield")
24
```

- Use an underscore _ to omit an argument label
- Question: Why so many options?

SWIFT ROADMAP

- [x] Variables
- [x] Control Flow
- [x] Functions
- [ ] **Classes**
- [ ] Structs
- [ ] Enums
- [ ] Optionals

CLASSES

- A `class` is a group of properties (state) and methods (behavior)
    - Properties are variables and constants defined on a class
    - Methods are functions defined on a class

CLASSES

```
class Book {
    let title: String
    let author: String
    let pages: Int

    var currentPage: Int = 0

    init(title: String, author: String, pages: Int) {
        self.title = title
        self.author = author
        self.pages = pages
    }

    func read(numPages: Int) {
        currentPage = min(currentPage + numPages, pages)
    }
}
```

Properties

Methods

CLASSES

- All stored properties must have a value when an instance of the class is created
- You may either:
  - Provide a default value when declaring the property
  - Set the property's value in an initializer

CLASSES

```
class Book {
    let title: String
    let author: String
    let pages: Int

    var currentPage: Int = 0

    init(title: String, author: String, pages: Int) {
        self.title = title
        self.author = author
        self.pages = pages
    }

    func read(numPages: Int) {
        currentPage = min(currentPage + numPages, pages)
    }
}
```

Property with a default value of zero

Properties set in an initializer

CLASSES

```
let book = Book(title: "Pride and Prejudice",   << Instantiate an object
                author: "Jane Austen",
                pages: 300)


book.read(numPages: 50)    << Call a function
print(book.currentPage)    << Access a property
```

- An object is an instance of a class
- Use the initializer to instantiate an object
- Use dot syntax to access properties and methods on an object

CLASSES

```
class Book {
    let title: String
    let author: String
    let pages: Int

    var currentPage: Int = 0

    var finished: Bool {
        return currentPage == pages
    }

    // initializer, methods, etc.
```

Stored properties

Computed property

- Properties can be either stored or computed
- What is the difference?

CLASSES

```
class PictureBook: Book {
    let illustrator: String

    init(title: String, author: String, illustrator: String, pages: Int) {
        self.illustrator = illustrator
        super.init(title: title, author: author, pages: pages)
    }
}
```

- A subclass of another class inherits the properties and methods of its base class. This is called inheritance.
- A subclass can have additional properties and methods of its own

CLASSES

```
class Book {

    // properties, initializer, methods…

    func printSummary() {
        print("\(title) by \(author) (\(pages) pages)")
    }
}
```

```
class PictureBook: Book {

    // properties, initializer, methods…

    override func printSummary() {
        print("\(title) by \(author), illustrations by \(illustrator) (\(pages) pages)")
    }
}
```

- A subclass can override methods from its base class

CLASSES

```
class PictureBook: Book {

    // properties, initializer, methods…

    override func printSummary() {
        super.printSummary()
        print("Illustrations by \(illustrator)")
    }
}
```

- Use `super` to access properties or methods on the base class

SWIFT ROADMAP

- [x] Variables
- [x] Control Flow
- [x] Functions
- [x] Classes
- [ ] **Structs**
- [ ] Enums
- [ ] Optionals

STRUCTS

```
struct Address {
    let street: String
    let city: String
    let state: String
    let zip: String

    func formattedAddress() -> String {
        return """
               \(street)
               \(city), \(state)
               \(zip)
               """
    }
}
```

- Structs are similar to classes. Both have properties and methods.

STRUCTS

```
struct Address {
    let street: String
    let city: String
    let state: String
    let zip: String

    func formattedAddress() -> String {
        return """
               \(street)
               \(city), \(state)
               \(zip)
               """
    }
}
```

- However, there are some differences. What appears to be missing from this struct?

STRUCTS

```
struct Address {
    let street: String
    let city: String
    let state: String
    let zip: String
}

let address = Address(street: "5730 S Ellis Ave",
                      city: "Chicago",
                      state: "IL",
                      zip: "60637")
```

- An initializer is created automatically with all properties that do not have a default value
- This is called a memberwise initializer

STRUCTS

```
struct Address {
    let street: String
    let city: String
    let state: String
    let zip: String

    init(street: String, city: String, state: String) {
        self.street = street
        self.city = city
        self.state = state
        self.zip = ZipUtil.lookup(street: street, city: city, state: state)
    }
}
```

- If you add your own initializer to a struct, the memberwise initializer will **not** be created.

STRUCTS

```
struct BusinessAddress: Address {
    let company: String
}
```

(This won't work!)

- The second key difference between structs and classes is that structs **do not** support inheritance

STRUCTS

- The third key difference is that structs are **copied** when they are passed around in your code
  - Structs: Pass by value
  - Classes: Pass by reference

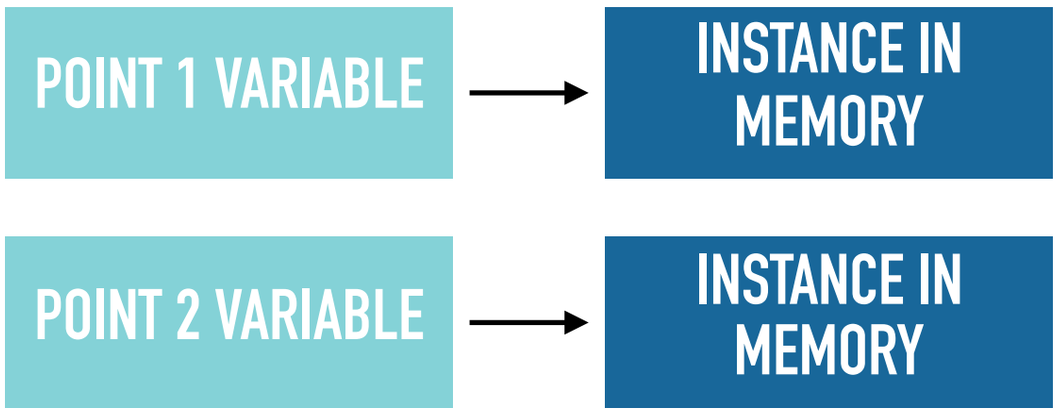
STRUCTS

```
struct Point {
    var x: Int
    var y: Int
}

var point1 = Point(x: 0, y: 0)
var point2 = point1

point2.x = 50
point2.y = 50

print("(\(point1.x), \(point1.y))")
```

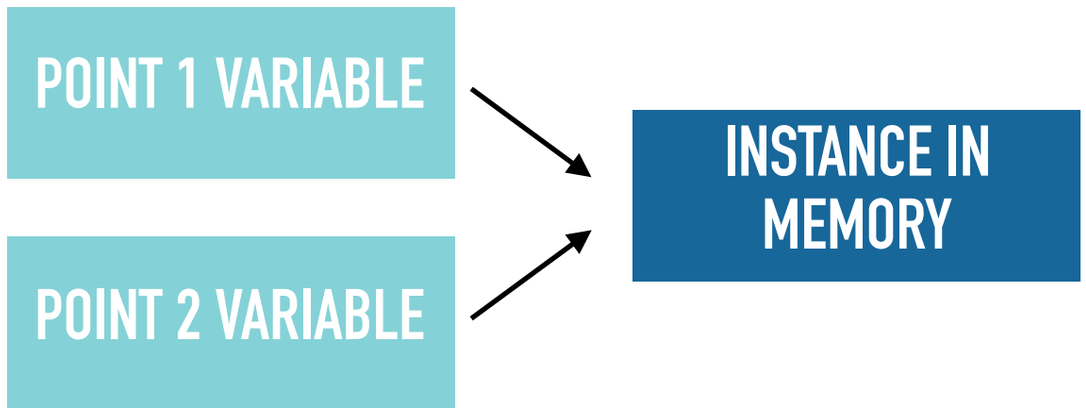
- What is printed?
- What's printed if you make `Point` a class?

STRUCTS

Struct (Pass by value)



Class (Pass by reference)



STRUCTS

```
struct Shape {
    var center: Point
}

let initialPoint = Point(x: 5, y: 5)
var circle = Shape(center: initialPoint)
var square = Shape(center: initialPoint)

square.center.x = 2
```

- Consider the code above. Would you want `Point` to be a struct or a class?

SWIFT ROADMAP

- [x] Variables
- [x] Control Flow
- [x] Functions
- [x] Classes
- [x] Structs
- [ ] **Enums**
- [ ] Optionals

ENUMS

```
enum Direction {
    case north
    case south
    case east
    case west
}
```

- Enums allow you to define a list of possible values.

ENUMS

```
enum Direction {

    case north, south, east, west

    var witch: String {
        switch self {
        case .north:
            return "Good Witch of the North"
        case .south:
            return "Glinda the Good Witch"
        case .east:
            return "Wicked Witch of the East"
        case .west:
            return "Wicked Witch of the West"
        }
    }
}
```

- Like classes and structs, enums can have methods and computed properties (but not stored properties).

ENUMS

```
enum Direction {

    case north, south, east, west

    var witch: String {
        switch self {
        case .north:
            return "Good Witch of the North"
        case .south:
            return "Glinda the Good Witch"
        case .east:
            return "Wicked Witch of the East"
        case .west:
            return "Wicked Witch of the West"
        }
    }
}
```

- You can use a switch statement to switch over the cases of an enum.

ENUMS

```
func printWarning(for direction: Direction) {
    print("You may encounter: \(direction.witch)")
}

printWarning(for: Direction.west)
```

- This is an example of how you might use an enum
- You can omit the type (`.west` instead of `Direction.west`)
    - This only works if the compiler can infer the type

ENUMS

```
func printWarning(for direction: Direction) {
    print("You may encounter: \(direction.witch)")
}

printWarning(for: Direction.west)
```

- Why is using an enum safer than using a string?

SWIFT ROADMAP

- Variables
- Control Flow
- Functions
- Classes
- Structs
- Enums
- **Optionals**

OPTIONALS

- “A type that represents either a wrapped value or `nil`, the absence of a value.” - Swift Documentation

Optional with a Value

Empty Optional

OPTIONALS

- Programming languages typically use `nil` or `null` to represent the absence of a value

OPTIONALS

- Example: An app allows users to upload profile photos. Some users will upload photos, and others will not.

```
user1.photo = 🖼

user2.photo = nil
```

OPTIONALS

- Imagine you're designing a programming language. What should happen if a developer tries to access the properties or methods of a `nil` photo?

```
user2.photo.pngData()
```

OPTIONALS

- What should happen if a developer tries to access the properties or methods of a `nil` photo?
    - Option 1: Do nothing (no-op)
    - Option 2: Terminate the application (crash)
- What are the pros and cons of each approach?

OPTIONALS

- Swift addresses this dilemma in two ways.
    - 1) The compiler can guarantee that certain values will never be `nil`. These are called non-optionals.
- 2) When working with optionals, developers choose between a no-op and a crash if the value is `nil`.

OPTIONALS

```
class User {
    var username: String
    var photo: UIImage?

    // initializer, methods, etc…
}
```

- By default, types are non-optional (e.g., `String`)
- A question mark indicates that a type is optional (e.g., `UIImage?`)

OPTIONALS

```
let bytes = user.photo?.pngData()?.count
```

- Use a question mark to access properties and methods on an optional
- This is called optional chaining
- If either `photo` or `pngData()` is nil, then `bytes` will be nil

OPTIONALS

```
let bytes = user.photo!.pngData()!.count
```

- Or use an exclamation mark to access properties and methods on an optional
- This is called forced unwrapping
- If either `photo` or `pngData()` is nil, the code will crash

OPTIONALS

```
let bytes = user.photo?.pngData()?.count ?? 0
```

- Use `??` to provide a fallback value
- This is called nil coalescing
- If either `photo` or `pngData()` is nil, then `bytes` will be zero

OPTIONALS

```
if let data = user.photo?.pngData() {
    print(data.count) // data has a value
}
```

- Use `if let` to unwrap an optional and, if a value is present, assign it to a new variable
- This is called optional binding

OPTIONALS

```
guard let data = user.photo?.pngData() else {
    return
}
```

- Or use `guard let` to unwrap an optional and, if a value is present, assign it to a new variable
- This is also optional binding

OPTIONALS

- When would you use `if let`?
- When would you use `guard let`?

SWIFT ROADMAP

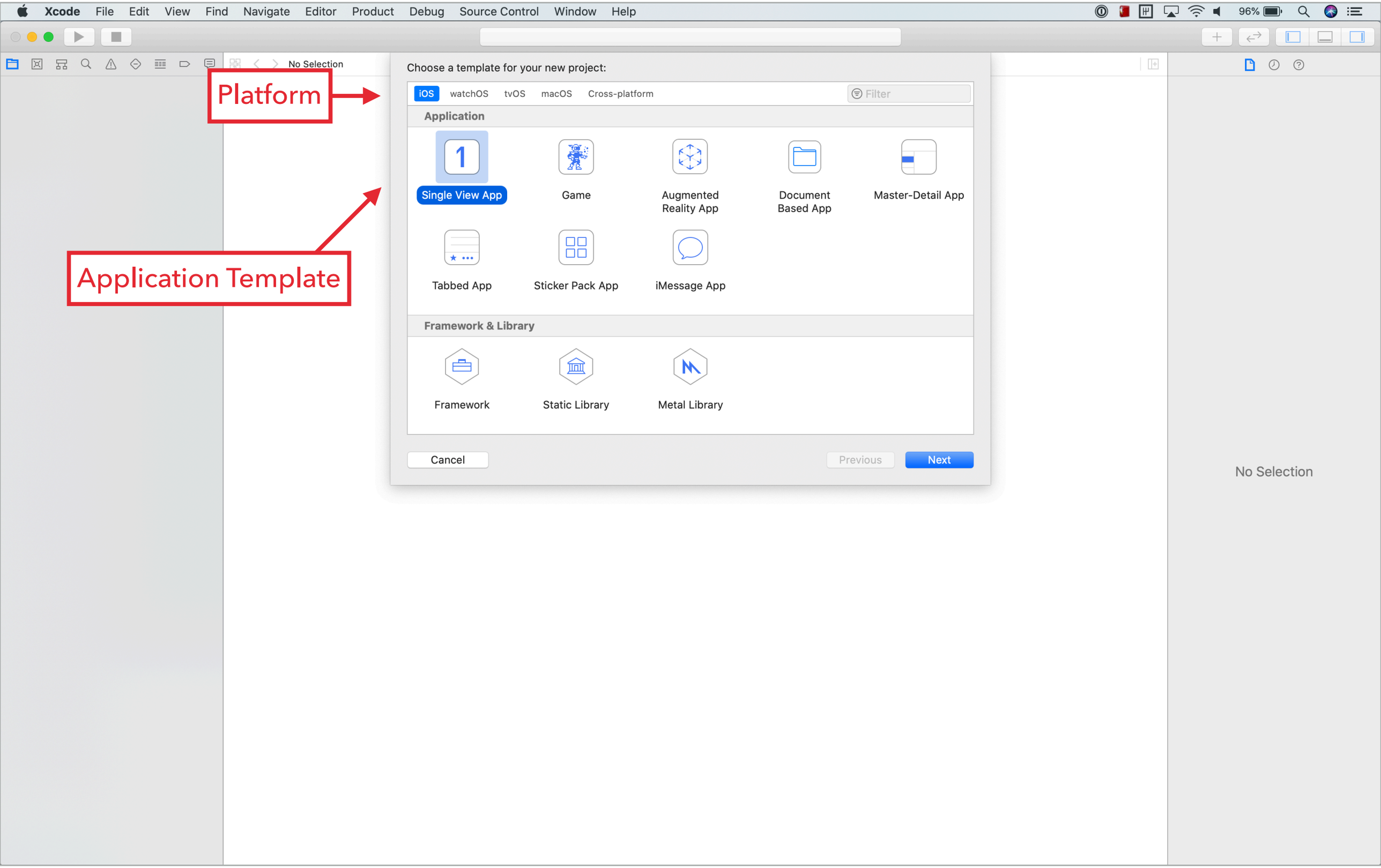
- Variables
- Control Flow
- Functions
- Classes
- Structs
- Enums
- Optionals

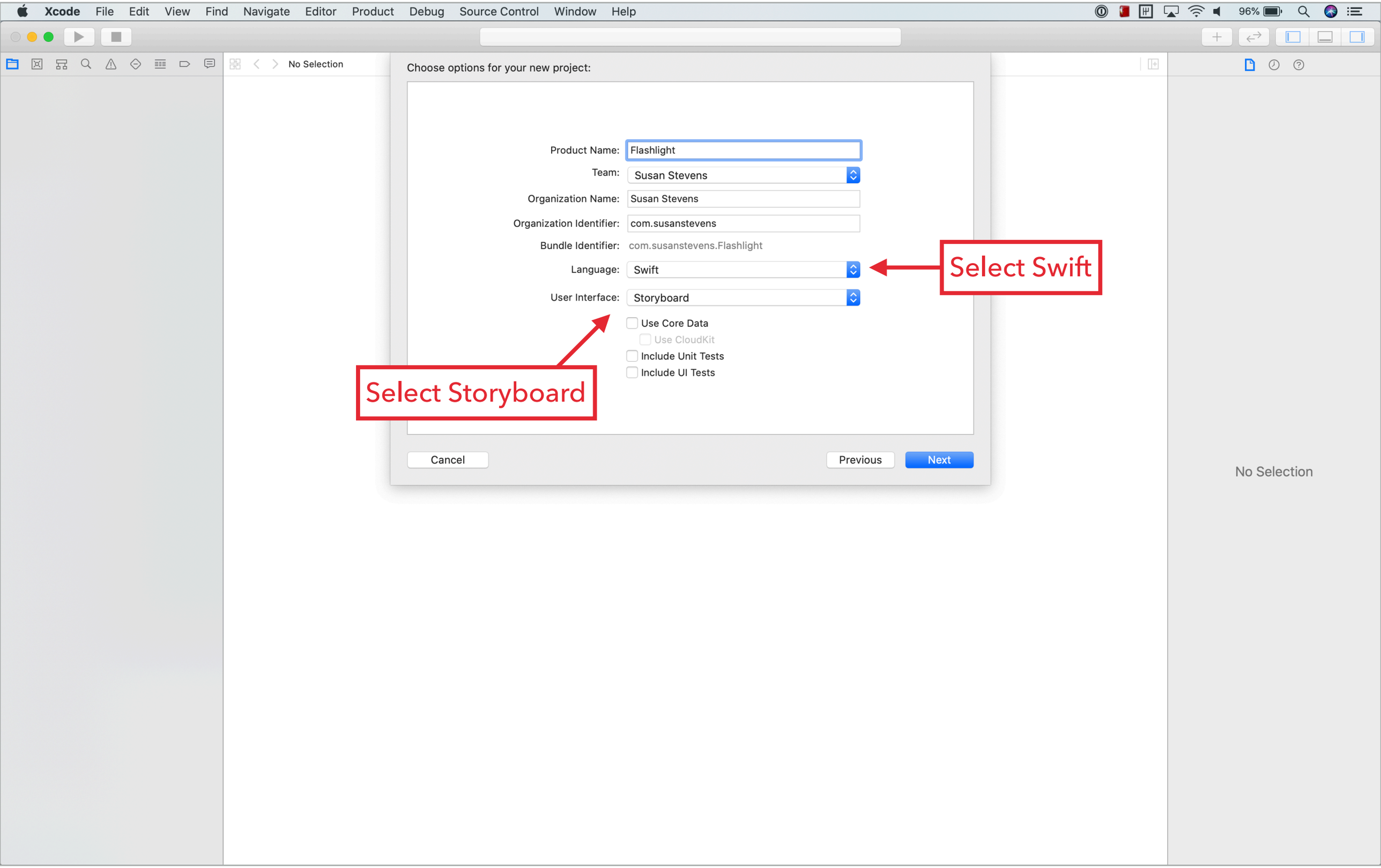
LET'S BUILD A

FLASHLIGHT

SAY HELLO 👋 TO

XCODE 


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
96%
No Selection
Platform
Application Template
Choose a template for your new project:
iOS watchOS tvOS macOS Cross-platform
Filter
Application
Single View App
Game
Augmented Reality App
Document Based App
Master-Detail App
Tabbed App
Sticker Pack App
iMessage App
Framework & Library
Framework
Static Library
Metal Library
Cancel
Previous
Next
No Selection


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
96%
No Selection
Choose options for your new project:
Product Name: Flashlight
Team: Susan Stevens
Organization Name: Susan Stevens
Organization Identifier: com.susanstevens
Bundle Identifier: com.susanstevens.Flashlight
Language: Swift
User Interface: Storyboard
Use Core Data
Use CloudKit
Include Unit Tests
Include UI Tests
Select Swift
Select Storyboard
Cancel
Previous
Next
No Selection

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

91%

Flashlight 〉 iPhone 11 Pro Max

Flashlight: **Ready** | Today at 11:10 AM

Toolbar

Flashlight 〉 Flashlight 〉 ViewController.swift 〉 No Selection

- Flashlight M
  - Flashlight
    - AppDelegate.swift A
    - SceneDelegate.swift A
    - ViewController.swift A
    - Main.storyboard A
    - Assets.xcassets M
    - LaunchScreen.storyboard A
    - Info.plist A
  - Products

```
1  //
2  //  ViewController.swift
3  //  Flashlight
4  //
5  //  Created by Susan Stevens on 1/2/20.
6  //  Copyright © 2020 Susan Stevens. All rights reserved.
7  //
8
9  import UIKit
10
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view.
16     }
17
18
19 }
20
21
```

Editor

Identity and Type

Name: ViewController.swift

Type: Default - Swift Source

Location: Relative to Group

ViewController.swift

Full Path: /Users/susanstevens/Documents/UChicago/iOS-Winter-2020/MPCS-51030-Winter-2020/sample-code/Session1/Flashlight/Flashlight/ViewController.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

Flashlight

Text Settings

Text Encoding: No Explicit Encoding

Line Endings: No Explicit Line Endings

Indent Using: Spaces

Widths: 4 (Tab) 4 (Indent)

Wrap lines

Navigator

Utility Area

Filter

Auto

Filter

All Output

Filter

Debug Area

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

Flashlight 〉 iPhone 11 Pro Max

Flashlight: **Ready** | Today at 11:10 AM

Flashlight 〉 Flashlight 〉 ViewController.swift 〉 No Selection

- Flashlight M
  - Flashlight
    - AppDelegate.swift A
    - SceneDelegate.swift A
    - ViewController.swift A
    - Main.storyboard A
    - Assets.xcassets M
    - LaunchScreen.storyboard A
    - Info.plist A
  - Products

```
1  //
2  //  ViewController.swift
3  //  Flashlight
4  //
5  //  Created by Susan Stevens on 1/2/20.
6  //  Copyright © 2020 Susan Stevens. All rights reserved.
7  //
8
9  import UIKit
10
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view.
16     }
17
18
19 }
20
21
```

Show/Hide Navigator, Debug Area and Utility Area

Identity and Ty

Name: ViewController.swift

Type: Default - Swift Source

Location: Relative to Group

ViewController.swift

Full Path: /Users/susanstevens/Documents/UChicago/iOS-Winter-2020/MPCS-51030-Winter-2020/sample-code/Session1/Flashlight/Flashlight/ViewController.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

Flashlight

Text Settings

Text Encoding: No Explicit Encoding

Line Endings: No Explicit Line Endings

Indent Using: Spaces

Widths: 4 Tab, 4 Indent

Wrap lines

Navigator

Utility Area

Debug Area

Auto

Filter

All Output

Filter

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

Flashlight 〉 iPhone 11 Pro Max | Finished running Flashlight on iPhone 11 Pro Max

Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard A
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products

Xcode Project File

Deployment Info

Flashlight

General | Signing & Capabilities | Resource Tags | Info | Build Settings | Build Phases | Build Rules

PROJECT
Flashlight
TARGETS
Flashlight

Identity

Display Name: Flashlight
Bundle Identifier: com.susanstevens.Flashlight
Version: 1.0
Build: 1

Deployment Info

Target	Device
iOS 13.2	iPhone
	iPad
	Mac (requires macOS 10.15)

Main Interface: Main

Device Orientation: Portrait, Upside Down, Landscape Left, Landscape Right

Status Bar Style: Default

Hide status bar
Requires full screen
Supports multiple windows

App Icons and Launch Images

App Icons Source: AppIcon
Launch Screen File: LaunchScreen

Frameworks, Libraries, and Embedded Content

Name	Embed

Add frameworks, libraries, and embedded content here

Development Assets

Add development assets here

Add folders, groups, or individual assets for use during development

Filter

Identity and Type

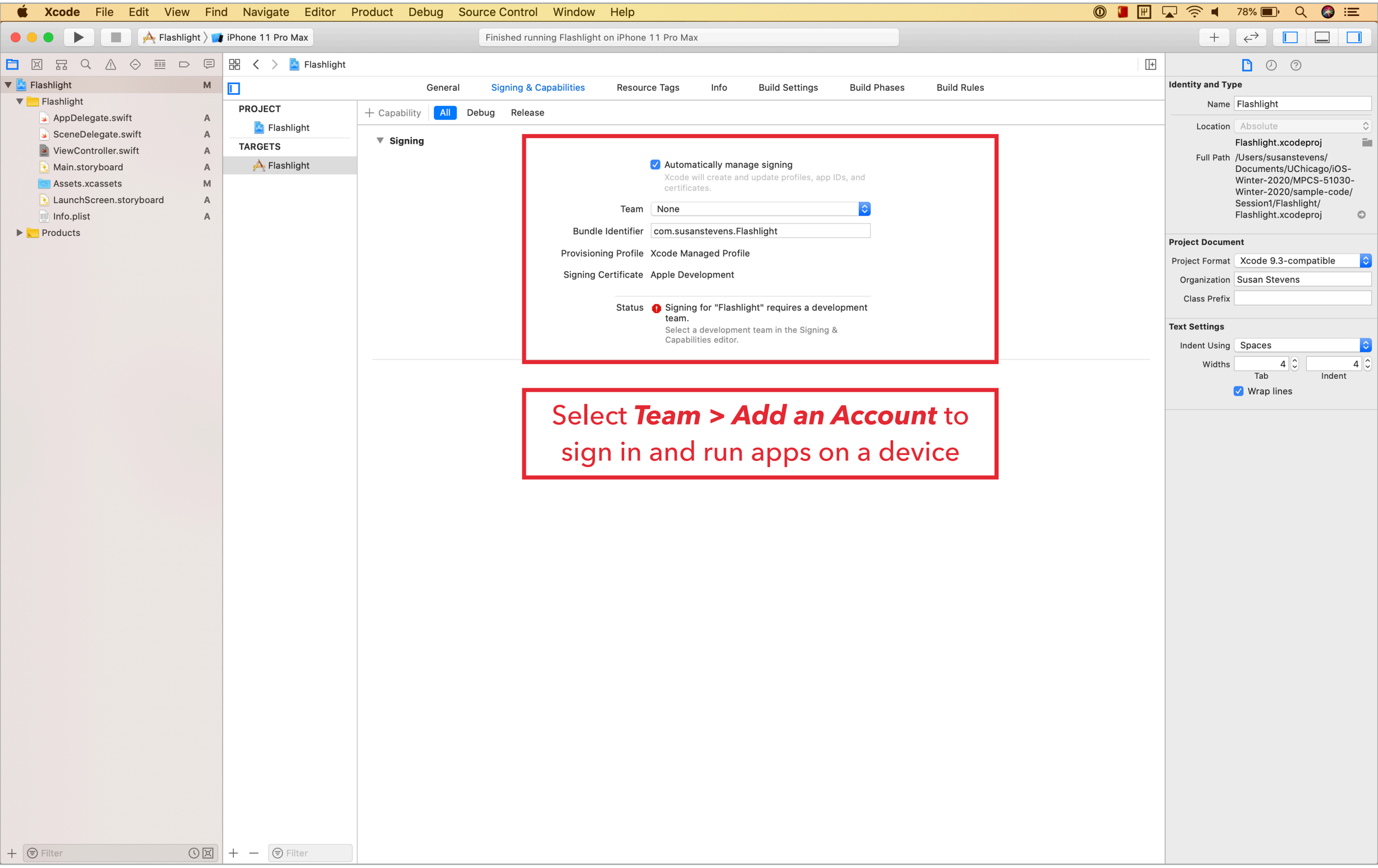
Name: Flashlight
Location: Absolute
Flashlight.xcodeproj
Full Path: /Users/susanstevens/Documents/UChicago/iOS-Winter-2020/MPCS-51030-Winter-2020/sample-code/Session1/Flashlight/Flashlight.xcodeproj

Project Document

Project Format: Xcode 9.3-compatible
Organization: Susan Stevens
Class Prefix:

Text Settings

Indent Using: Spaces
Widths: 4 Tab, 4 Indent
Wrap lines


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
78%
Flashlight 〉 iPhone 11 Pro Max
Finished running Flashlight on iPhone 11 Pro Max
Flashlight
Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard A
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products
Filter
Flashlight
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
Flashlight
TARGETS
Flashlight
Capability
All
Debug
Release
Signing
Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.
Team None
Bundle Identifier com.susanstevens.Flashlight
Provisioning Profile Xcode Managed Profile
Signing Certificate Apple Development
Status Signing for "Flashlight" requires a development team.
Select a development team in the Signing & Capabilities editor.
Select Team > Add an Account to sign in and run apps on a device
Identity and Type
Name Flashlight
Location Absolute
Flashlight.xcodeproj
Full Path /Users/susanstevens/Documents/UChicago/iOS-Winter-2020/MPCS-51030-Winter-2020/sample-code/Session1/Flashlight/Flashlight.xcodeproj
Project Document
Project Format Xcode 9.3-compatible
Organization Susan Stevens
Class Prefix
Text Settings
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
Filter
Filter

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

83%

Flashlight 〉 iPhone 11 Pro Max | Running Flashlight on iPhone 11 Pro Max

Flashlight 〉 Flashlight 〉 ViewController.swift 〉 No Selection

- Flashlight M
  - Flashlight
    - AppDelegate.swift A
    - SceneDelegate.swift A
    - ViewController.swift A
    - Main.storyboard A
    - Assets.xcassets M
    - LaunchScreen.storyboard A
    - Info.plist A
  - Products

```
1  //
2  //  ViewController.swift
3  //  Flashlight
4  //
5  //  Created by Susan Stevens on 1/2/20.
6  //  Copyright © 2020 Susan Stevens. All rights reserved.
7  //
8
9
10
11
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view.
16     }
17
18
19 }
20
21
```

Select simulator or device

Run and Stop

Flashlight

Auto | Filter | All Output | Filter

11:37

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

58%

Flashlight 〉 iPhone 11 Pro Max

Running Flashlight on iPhone 11 Pro Max

Flashlight 〉 Flashlight 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 View Controller Scene 〉 View Controller

Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard A
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products

Storyboard

View Controller Scene
View Controller
View
Safe Area
First Responder
Exit
Storyboard Entry Point

Identity Inspector

Custom Class
Class ViewController
Module Flashlight
Inherit Module From Target

Identity
Storyboard ID
Restoration ID
Use Storyboard ID

User Defined Runtime Attributes
Key Path | Type | Value

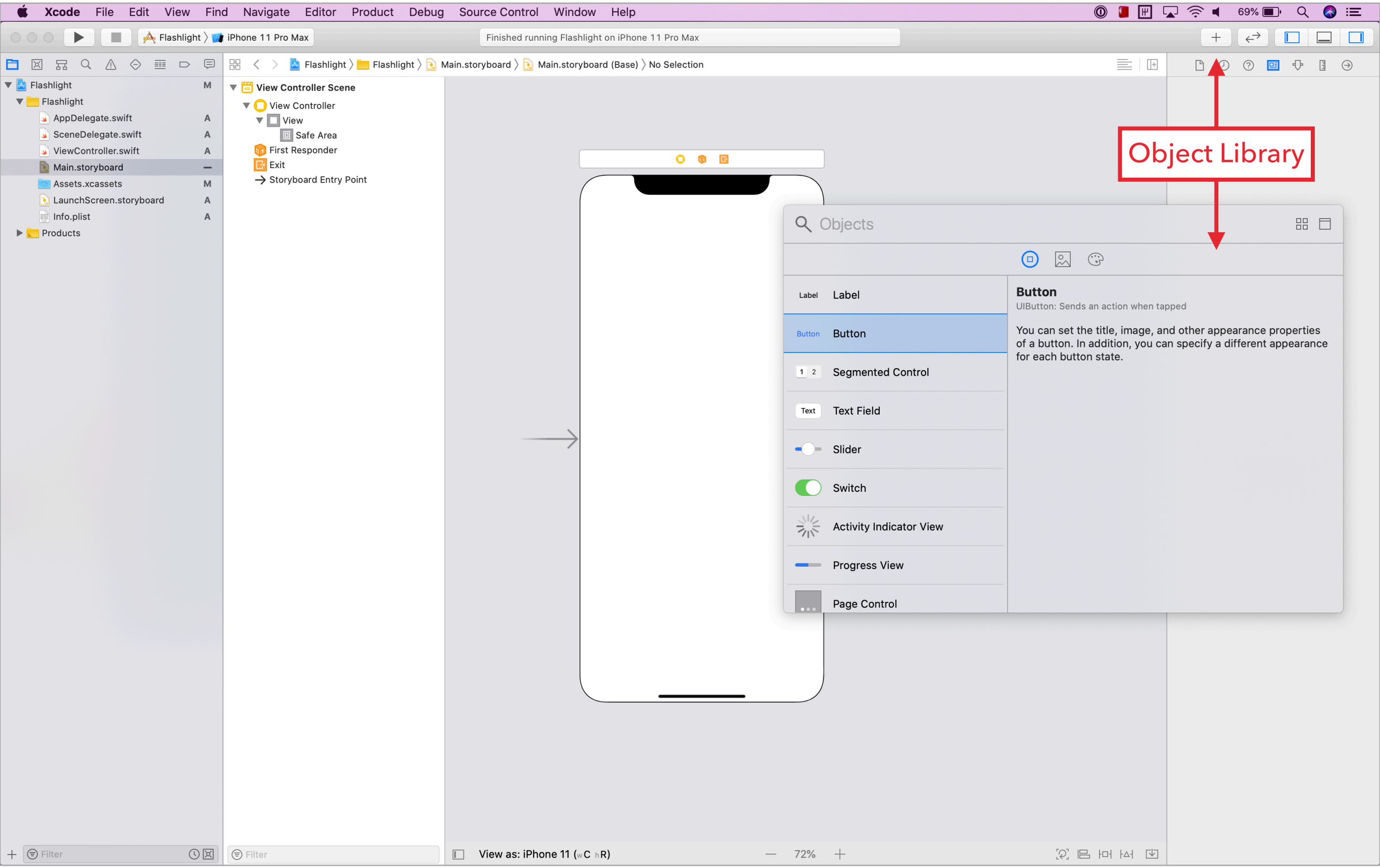
Document
Label Xcode Specific Label
Object ID BYZ-38-t0r
Lock Inherited - (Nothing)
Localizer Hint Comment For Localizer

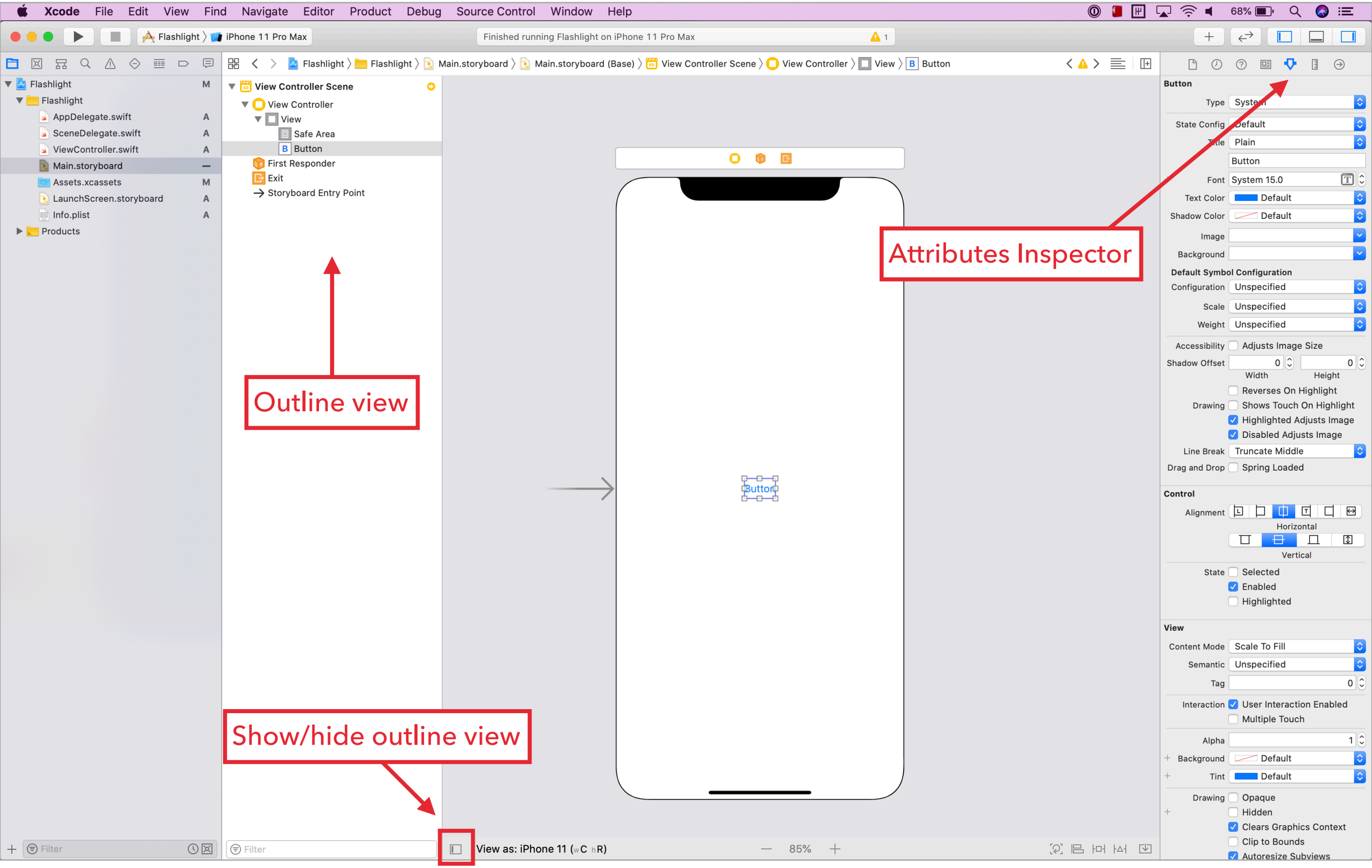
Filter

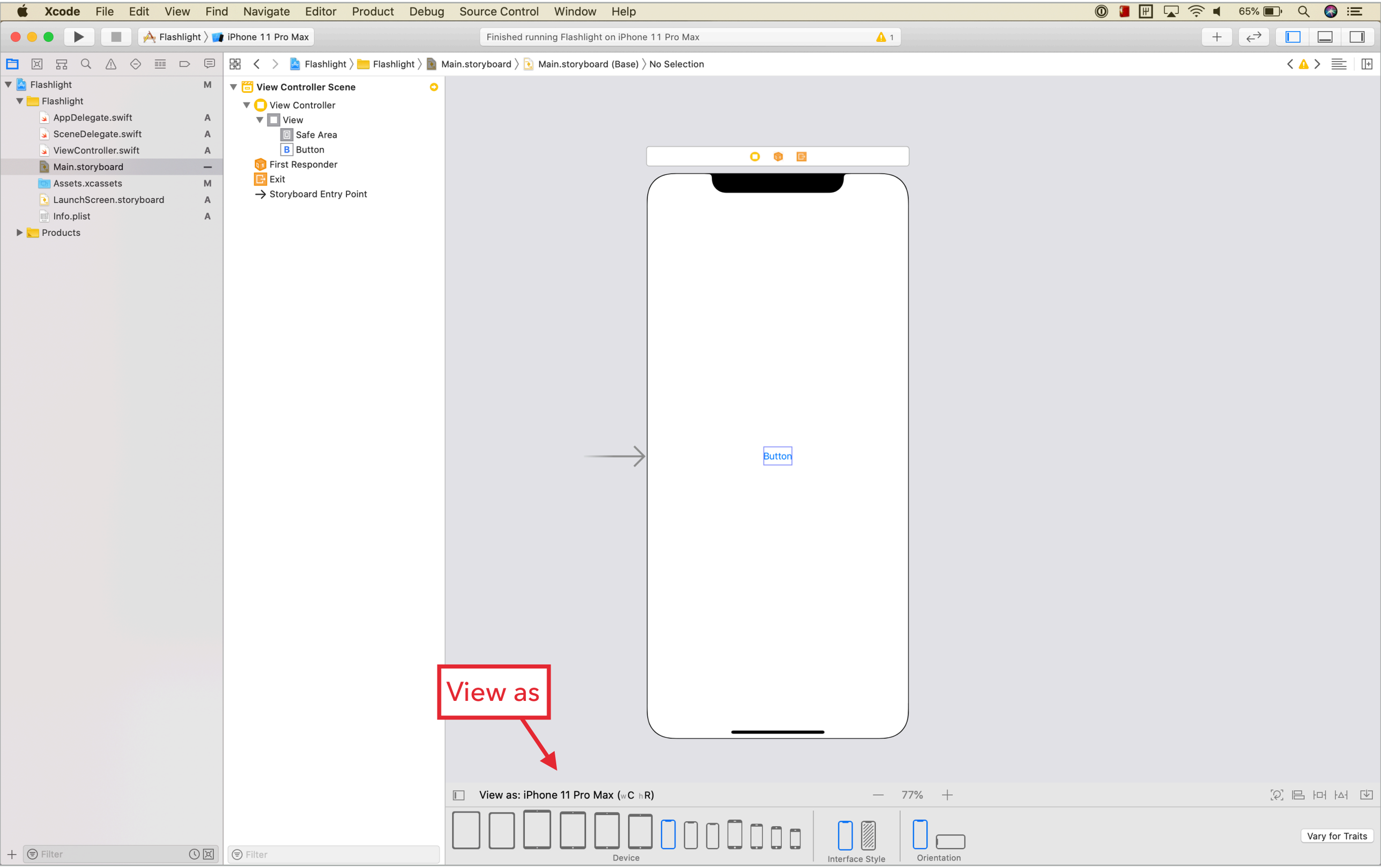
View as: iPhone 11 Pro Max (wC hR) 78%

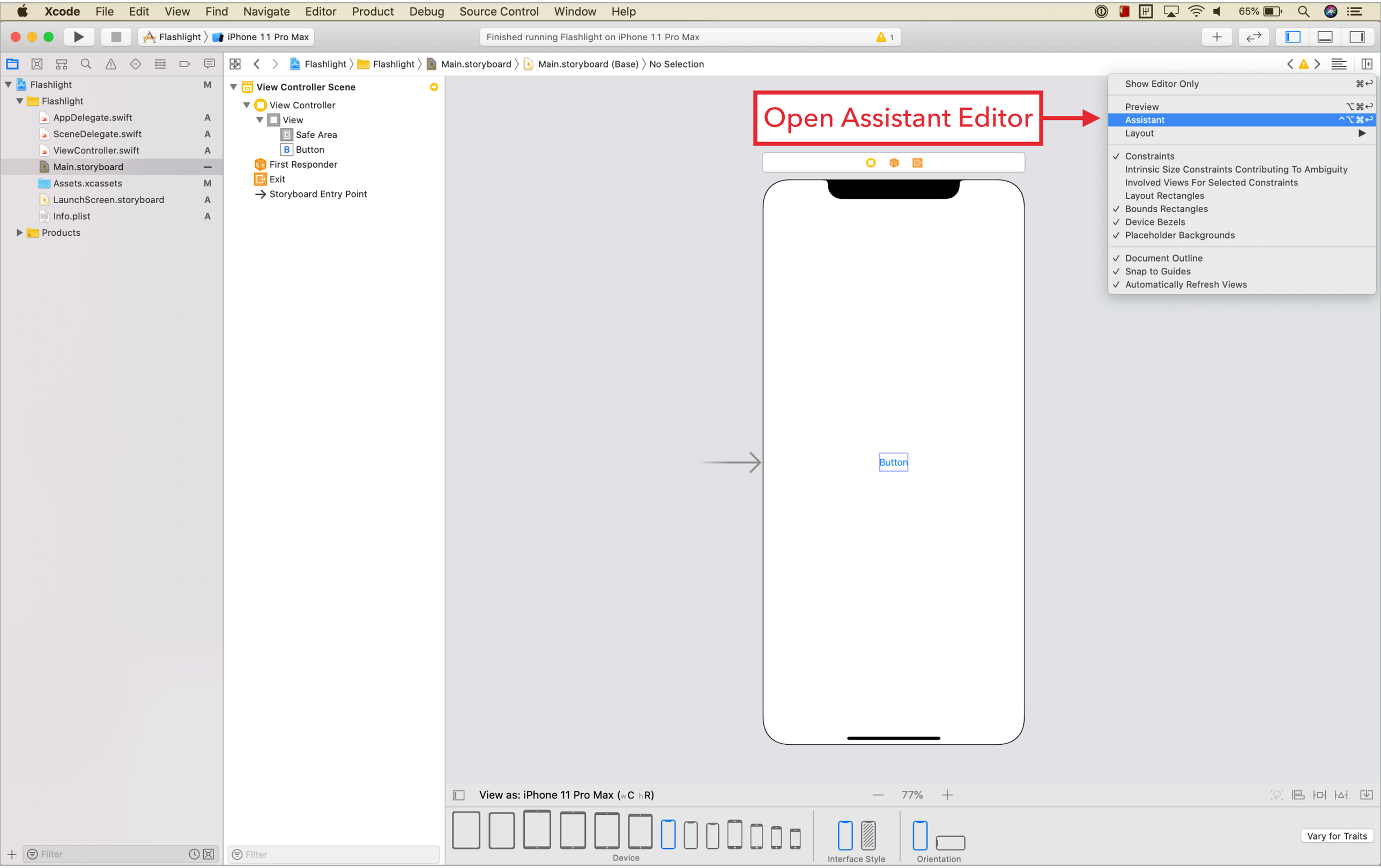
Filter

Flashlight


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
69%
Flashlight 〉 iPhone 11 Pro Max
Finished running Flashlight on iPhone 11 Pro Max
Flashlight 〉 Flashlight 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection
Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard —
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products
View Controller Scene
View Controller
View
Safe Area
First Responder
Exit
Storyboard Entry Point
Object Library
Objects
Label Label
Button Button
1 2 Segmented Control
Text Text Field
Slider
Switch
Activity Indicator View
Progress View
Page Control
Button
UIButton: Sends an action when tapped
You can set the title, image, and other appearance properties of a button. In addition, you can specify a different appearance for each button state.
Filter
Filter
View as: iPhone 11 (wC hR)
72%


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Flashlight 〉 iPhone 11 Pro Max
Finished running Flashlight on iPhone 11 Pro Max
Flashlight 〉 Flashlight 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 View Controller Scene 〉 View Controller 〉 View 〉 Button
Flashlight
Flashlight
AppDelegate.swift
SceneDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
View Controller Scene
View Controller
View
Safe Area
Button
First Responder
Exit
Storyboard Entry Point
Outline view
Attributes Inspector
Show/hide outline view
Button
Type
State Config
Default
Title
Plain
Font
System 15.0
Text Color
Shadow Color
Image
Background
Default Symbol Configuration
Configuration
Unspecified
Scale
Weight
Accessibility
Adjusts Image Size
Shadow Offset
Width
Height
Reverses On Highlight
Drawing
Shows Touch On Highlight
Highlighted Adjusts Image
Disabled Adjusts Image
Line Break
Truncate Middle
Drag and Drop
Spring Loaded
Control
Alignment
Horizontal
Vertical
State
Selected
Enabled
Highlighted
View
Content Mode
Scale To Fill
Semantic
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Tint
Opaque
Hidden
Clears Graphics Context
Clip to Bounds
Autoresize Subviews
Filter
View as: iPhone 11 (wC hR)
85%


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
65%
Flashlight 〉 iPhone 11 Pro Max
Finished running Flashlight on iPhone 11 Pro Max
1
Flashlight 〉 Flashlight 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection
Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard —
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products
View Controller Scene
View Controller
View
Safe Area
Button
First Responder
Exit
Storyboard Entry Point
Button
View as
View as: iPhone 11 Pro Max (wC hR)
77%
Device
Interface Style
Orientation
Vary for Traits
Filter
Filter


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Flashlight 〉 iPhone 11 Pro Max
Finished running Flashlight on iPhone 11 Pro Max
Flashlight 〉 Flashlight 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection
Flashlight
Flashlight
AppDelegate.swift
SceneDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
View Controller Scene
View Controller
View
Safe Area
Button
First Responder
Exit
Storyboard Entry Point
Open Assistant Editor
Show Editor Only
Preview
Assistant
Layout
Constraints
Intrinsic Size Constraints Contributing To Ambiguity
Involved Views For Selected Constraints
Layout Rectangles
Bounds Rectangles
Device Bezels
Placeholder Backgrounds
Document Outline
Snap to Guides
Automatically Refresh Views
Button
View as: iPhone 11 Pro Max (wC hR)
77%
Device
Interface Style
Orientation
Vary for Traits
Filter
Filter

Control+drag from storyboard to code

```
//
//  ViewController.swift
//
//  Created by Susan Stevens on 1/2/20.
//  Copyright © 2020 Susan Stevens. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }


}
```

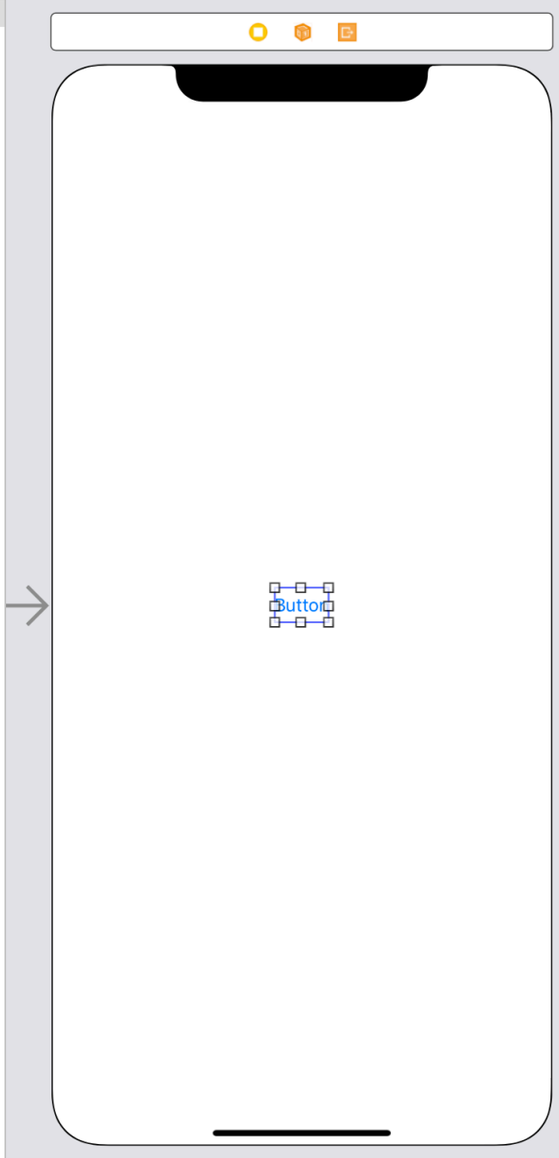
Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

Flashlight 〉 iPhone 11 Pro Max | Running Flashlight on iPhone 11 Pro Max | 1

Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard —
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products

Flashlight 〉 Fl...ht 〉 M...rd 〉 M...e) 〉 Vi...e 〉 Vi...er 〉 View 〉 B Button

View Controller Scene
View Controller
View
Safe Area
Button
First Responder
Exit
Storyboard Entry Point



Filter

View as: iPhone 11 Pro Max (wC hR)

Automatic 〉 ViewController.swift 〉 No Selection

```swift
//
//  ViewController.swift
//  Flashlight
//
//  Created by Susan Stevens on 1/2/20.
//  Copyright © 2020 Susan Stevens. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func buttonPressed(_ sender: Any) {
        print("button pressed")
    }

}
```

Filled in circle means the function is hooked up to the button

Filter

Flashlight

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

Flashlight 〉 iPhone 11 Pro Max | Running Flashlight on iPhone 11 Pro Max

Flashlight
- Flashlight
  - AppDelegate.swift A
  - SceneDelegate.swift A
  - ViewController.swift A
  - Main.storyboard —
  - Assets.xcassets M
  - LaunchScreen.storyboard A
  - Info.plist A
- Products

View Controller Scene
- View Controller
  - View
    - Safe Area
    - Button
- First Responder
- Exit
- Storyboard Entry Point

"Reverse drag" from code to storyboard

Button

View as: iPhone 11 Pro Max (wC hR)

```swift
//
//  ViewController.swift
//  Flashlight
//
//  Created by Susan Stevens on 1/2/20.
//  Copyright © 2020 Susan Stevens. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func buttonPressed(_ sender: Any) {
        print("button pressed")
    }

}
```

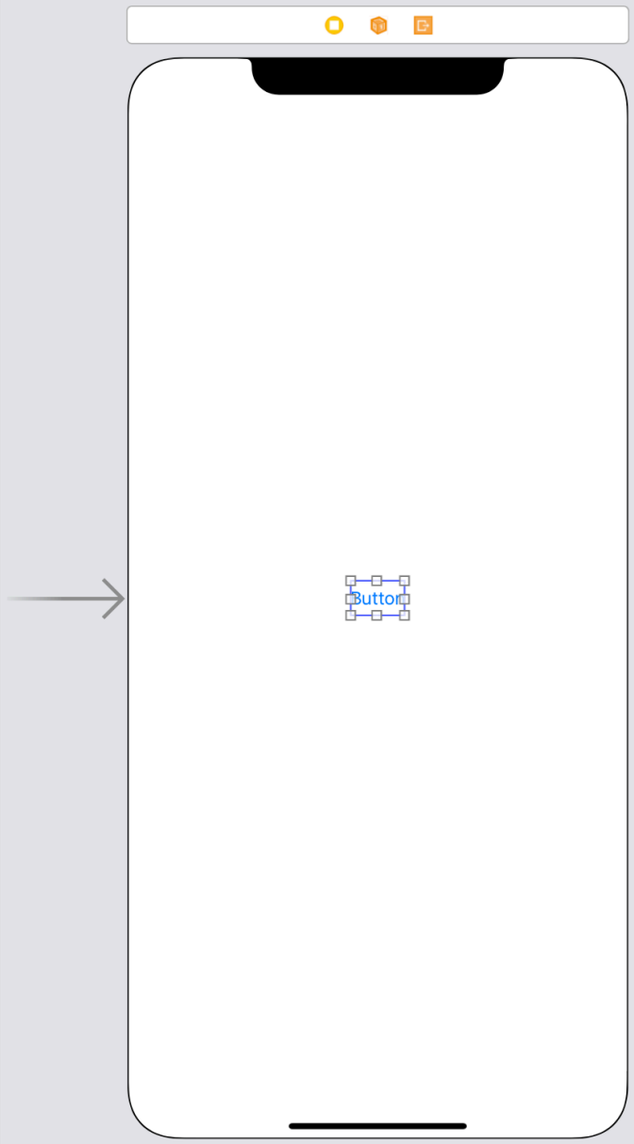
Flashlight

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

Flashlight › iPhone 11 Pro Max | Running Flashlight on iPhone 11 Pro Max

Flashlight › Fla...ht › M...ard › M...se) › Vi...ene › Vi...ller › View › Button

- View Controller Scene
  - View Controller
    - View
      - Safe Area
      - Button
    - First Responder
    - Exit
    - Storyboard Entry Point



View as: iPhone 11 Pro Max (wC hR)

Automatic › ViewController.swift › No Selection

```
1  //
2  //  ViewController.swift
3  //  Flashlight
4  //
5  //  Created by Susan Stevens on 1/2/20.
6  //  Copyright © 2020 Su
7  //
8
9  import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet var button: UIButton!
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17         // Do any additional setup after loading the view.
18     }
19
20     @IBAction func buttonPressed(_ sender: Any) {
21         print("button pressed")
22     }
23
24 }
25
26
```

Connections Inspector

Triggered Segues
- action

Outlet Collections
- gestureRecognizers

Sent Events
- Did End On Exit
- Editing Changed
- Editing Did Begin
- Editing Did End
- Primary Action Triggered
- Touch Cancel
- Touch Down
- Touch Down Repeat
- Touch Drag Enter
- Touch Drag Exit
- Touch Drag Inside
- Touch Drag Outside
- Touch Up Inside → View Controller buttonPressed:
- Touch Up Outside
- Value Changed

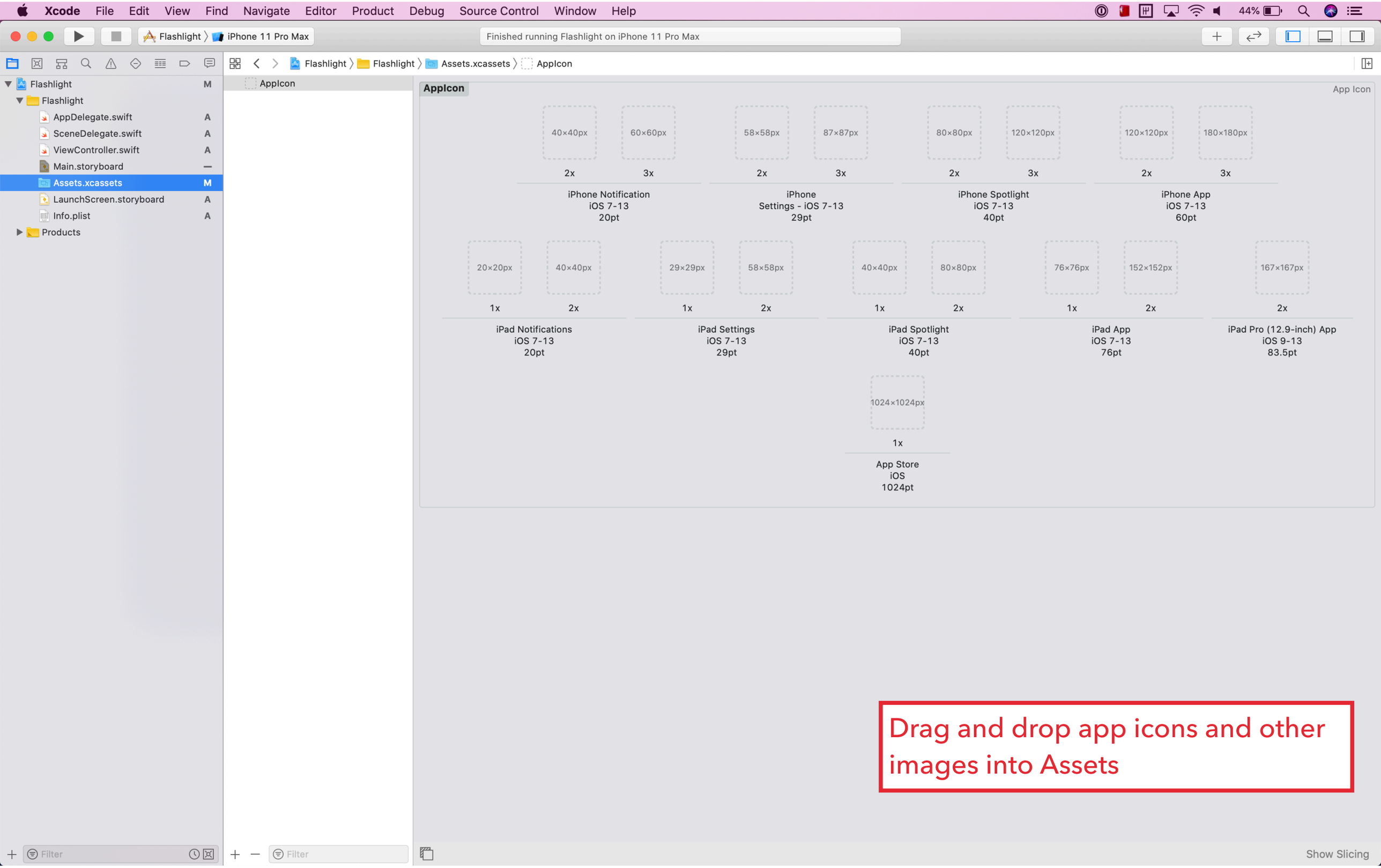
Referencing Outlets
- button → View Controller
- New Referencing Outlet

Referencing Outlet Collections
- New Referencing Outlet Collection

To delete a connection, you must remove it in the Connections Inspector, not just in code.

Filter

Flashlight


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
44%
Flashlight 〉 iPhone 11 Pro Max
Finished running Flashlight on iPhone 11 Pro Max
Flashlight 〉 Flashlight 〉 Assets.xcassets 〉 AppIcon
Flashlight M
Flashlight
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard —
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products
AppIcon
AppIcon
App Icon
40×40px
60×60px
2x
3x
iPhone Notification
iOS 7-13
20pt
58×58px
87×87px
2x
3x
iPhone
Settings - iOS 7-13
29pt
80×80px
120×120px
2x
3x
iPhone Spotlight
iOS 7-13
40pt
120×120px
180×180px
2x
3x
iPhone App
iOS 7-13
60pt
20×20px
40×40px
1x
2x
iPad Notifications
iOS 7-13
20pt
29×29px
58×58px
1x
2x
iPad Settings
iOS 7-13
29pt
40×40px
80×80px
1x
2x
iPad Spotlight
iOS 7-13
40pt
76×76px
152×152px
1x
2x
iPad App
iOS 7-13
76pt
167×167px
2x
iPad Pro (12.9-inch) App
iOS 9-13
83.5pt
1024×1024px
1x
App Store
iOS
1024pt
Drag and drop app icons and other images into Assets
Filter
Filter
Show Slicing

WORKING WITH

GIT AND GITHUB

WHAT IS VERSION CONTROL?

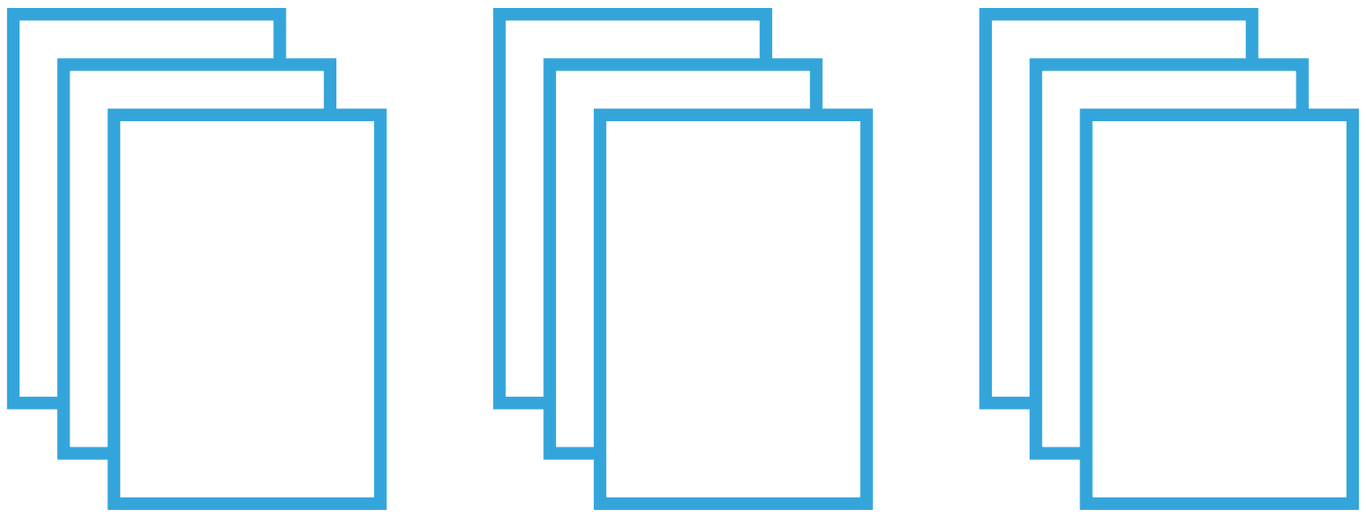
- Version control allows you to save and retrieve different snapshots (or versions) of a project as it evolves over time.
- Git is the most widely-used version control system

WHAT IS VERSION CONTROL?

- Why would you want to have multiple versions of a project?
- Think about:
  - Debugging
  - Building new features

BASIC TERMINOLOGY

- Repository (or repo): The collection of files and folders for a given project as well as each file's revision history

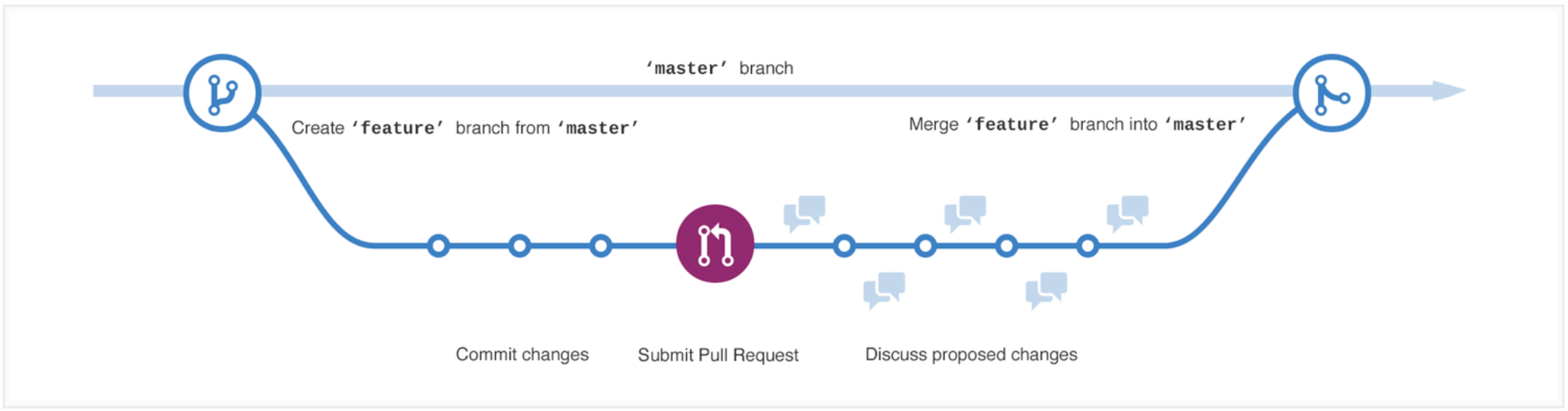


BASIC TERMINOLOGY

- Commit: A snapshot of the project at a particular point in time, along with:
  - Timestamp
  - Author
  - Commit message

BASIC TERMINOLOGY

- Branch: A separate line of development
- Branches allow developers to build out new features incrementally while maintaining a stable master branch



https://guides.github.com/activities/hello-world/

BASIC TERMINOLOGY

- Remote: A copy of the repository that is hosted on GitHub or another centralized place
- Developers work on local copies on the repo, and when they're ready, they push changes to GitHub

GIT COMMANDS

- Make a copy of a repository on your local machine:

```
$ git clone <github-url>
```

GIT COMMANDS

- Create a new branch:

```
$ git branch <branch-name>
```

- Switch to the new branch:

```
$ git checkout <branch-name>
```

GIT COMMANDS

- After making changes, stage the files that you want to include in the next commit:

```
$ git add README.md // stages README.md
```

```
$ git add . // stages all modified files
```

GIT COMMANDS

- Save a snapshot by making a commit:

```
$ git commit -m "Update README"
```

GIT COMMANDS

- View the status of your local repo (e.g., current branch, staged and unstaged files):

```
$ git status
```

GIT COMMANDS

- Push changes to GitHub:

```
$ git push
```

- Pull changes from GitHub:

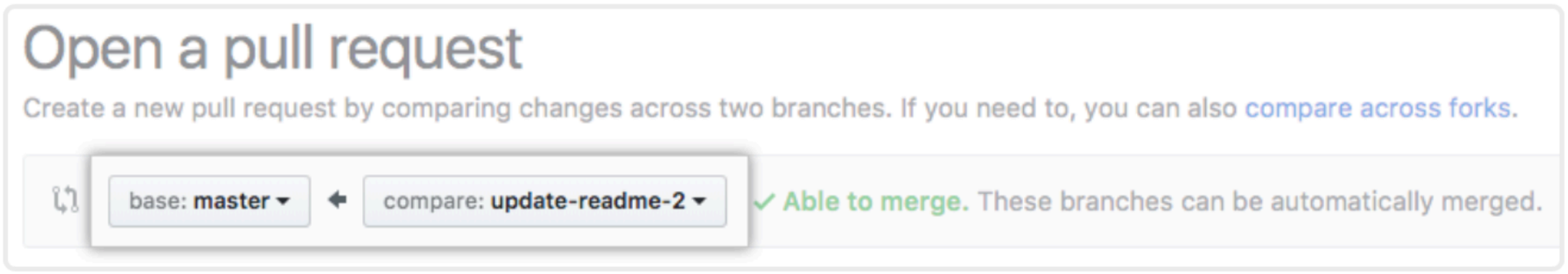
```
$ git pull
```

WORKING WITH GITHUB

- In addition to hosting remote repositories GitHub provides features like:
  - Issues (for questions/comments about a project)
  - Pull Requests (for discussion of proposed changes)
  - And more…

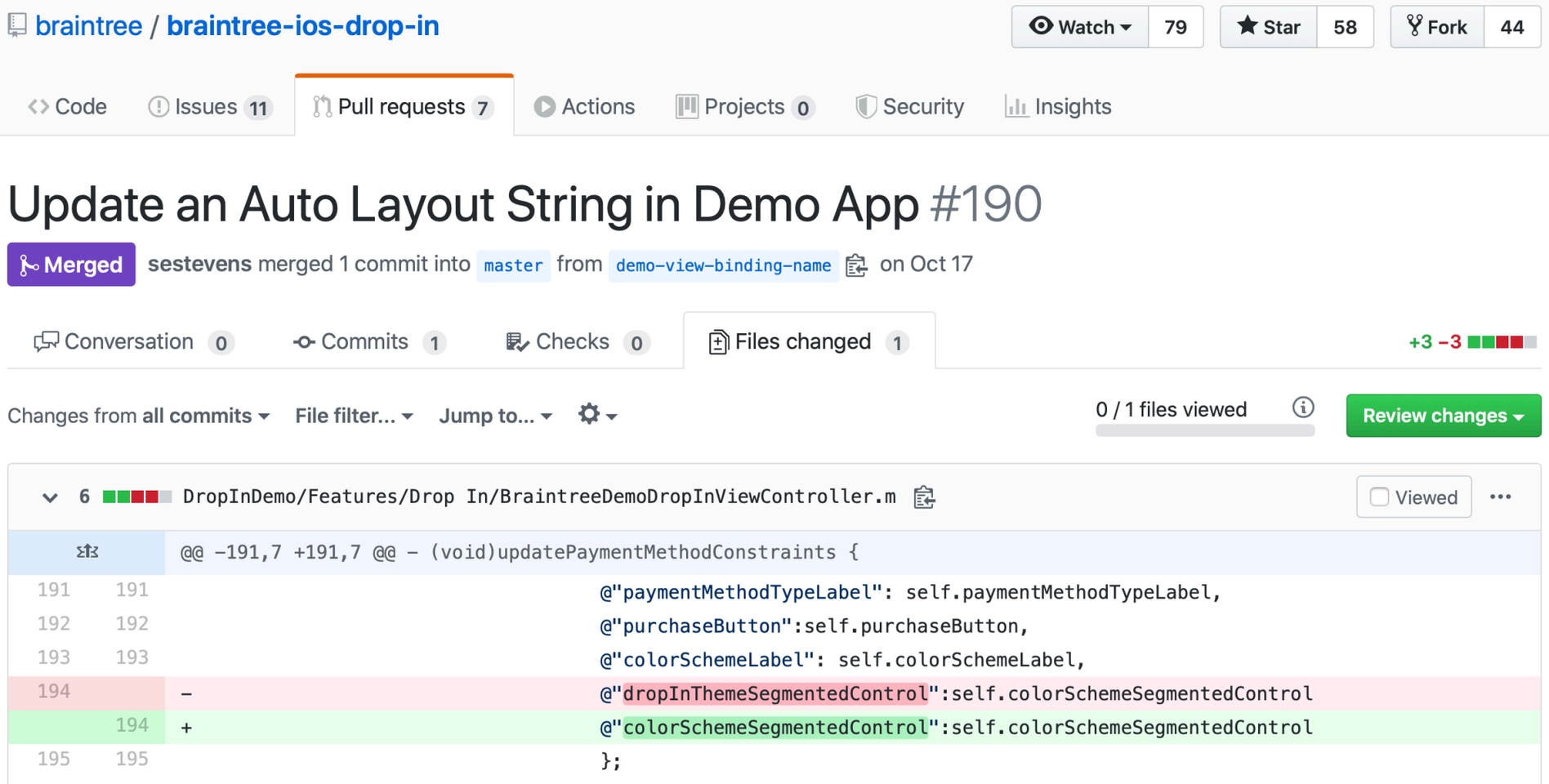
PULL REQUESTS

- When a new feature is ready to be merged into the master branch, the developer will open a pull request (or PR)



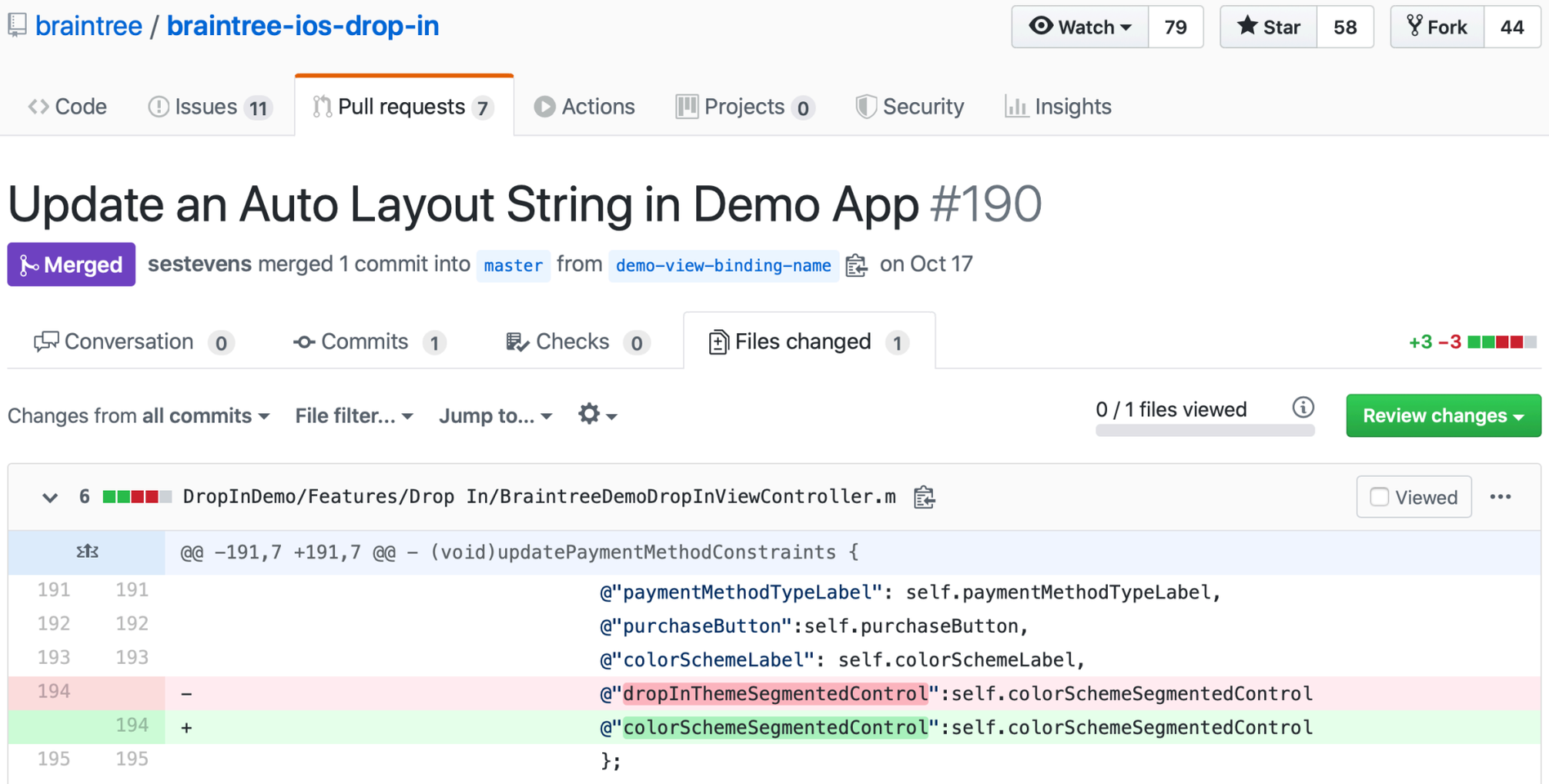
PULL REQUESTS

- This allows other developers to view a diff of the changes and add comments and suggestions



PULL REQUESTS

- Once the pull request has been approved, it can be merged into master



HOMEWORK ASSIGNMENTS

- For homework assignments, you will work on a branch called development
- When you're ready to submit, open a pull request from development into master
- We will add comments and your grade to the pull request

SESSION 1

REVIEW

REVIEW

- What was Steve Jobs' "sweet solution"?

REVIEW

- What are the three ways to create types in Swift?

REVIEW

- What are the key differences between classes and structs?

REVIEW

- What is the difference between Git and GitHub?

ASSIGNMENT #1

STORM VIEWER

BEFORE YOU START

- Create a GitHub account
- Create an Apple Developer account
- Download Xcode 11
- Bookmark the course website

REQUIREMENTS

- Fill out the student information form
- Post a message on Slack in the #general channel
- Download and read the “Swift Tour” playground
- Complete the “Storm Viewer” tutorial